150 ptas.



24

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen II - Fascículo 24

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones
Asesor técnico:	Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador,* se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-90-0 (tomo 2)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 278406
Impreso en España - Printed in Spain - Junio 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Apuestas por ordenador


Rex Features Ltd.

Un día en las carreras
Las carreras de caballos son un campo interesante para la aplicación de ordenadores personales, pero sólo para aquellas personas que saben lo que están haciendo. Así, por lo menos un criador utiliza un microordenador Apple II para llevar una base de datos con la raza de todos sus caballos, y valerse de ellos para intentar que salgan campeones

Los ordenadores ya tienen muchas aplicaciones en el mundo de las apuestas. Existen programas de quinielas incluso para ordenadores personales

Las apuestas se basan en la probabilidad, aunque a muchos jugadores les gustaría que se basaran en la certeza de ganar. Este deseo, por supuesto, carece de fundamento, dado que la inmensa mayoría de los jugadores pierden las más de las veces y, con frecuencia, grandes cantidades. Ello se debe a que las probabilidades están en contra de ellos y a favor del casino, del corredor de apuestas y del Patronato de Apuestas Mutuas. Para ver si los ordenadores pueden contribuir a compensar el equilibrio, primero debemos considerar estas probabilidades.

Despojados de sus adornos exteriores, todos los juegos de azar se reducen a apostar sobre un suceso aleatorio. Éste suele generarse mediante algún dispositivo, como una bola que gira dentro de la rueda de una ruleta, o una carta extraída de un mazo de naipes cuidadosamente barajado. Si se conocen los parámetros (la cantidad de naipes, p. ej.), la teoría de la probabilidad permite efectuar ciertas predicciones acerca de la verosimilitud de que el hecho fortuito se produzca. Por ejemplo, la ruleta que se utiliza en los casinos posee 37 hendiduras numeradas del 0 al 36. Existen, por lo tanto, 18 hendiduras de número impar y 18 hendiduras de número par donde puede caer la bola, más el cero. Las probabilidades de que la bola se deposite en una hendidura de número impar se pueden expresar como 18/37, o 0,4864864, o de un poco más del 48,6 %. Este porcentaje es ligeramente inferior a la probabilidad de que una moneda, después de lanzarla al aire, caiga del lado cara; la diferencia, determinada por la presencia de la hendidura del cero, representa la "ventaja" de la casa o su margen de beneficios.

Es esta "ventaja" lo que hace que los juegos de azar, y la mayoría de todas las otras formas de apuesta, sean tan poco gratificantes. A pesar de las argumentaciones que ocasionalmente aducen los proveedores de sistemas de apuestas, los ordenadores no pueden hacer nada por mejorar las probabilidades básicas de un juego determinado.

Tales objeciones teóricas no han conseguido desalentar a los inventores entusiastas y en la actualidad a los propietarios de ordenadores personales se les ofrece una variedad de sistemas de apuestas supuestamente a prueba de fallos, algunos de los cuales hasta parecen funcionar y todo. En el caso de los sistemas de apuestas para casino, éstos demuestran casi invariablemente ser variantes de la *duplicación*, un procedimiento que adolece del inconveniente de que para alcanzar el éxito hay que aportar una cantidad infinita de dinero al juego. Existe, asimismo, otro problema: aunque técnicamente no es ilegal, la administración de los casinos no permite usar ningún tipo de ordenador. Presumiblemente los ordenadores son de más ayuda cuando están implicadas la pericia y la estrategia.

Lamentablemente, la pericia exigida por la mayoría de los juegos de apuestas más populares es

mínima. Un caso concreto es el de las quinielas futbolísticas. En Gran Bretaña, el programa para predicción de quinielas más completo es el célebre *F4 Football Forecast*, del profesor Frank George, que está a la venta en versiones aptas para la mayoría de los ordenadores personales. Analizando las estadísticas de los últimos diez años, el programa le atribuye un valor al rendimiento promedio de cada equipo. Ajustando una línea para sopesar su situación a largo plazo (inferido a partir de las posiciones en las tablas de liga), su situación a corto plazo y el resultado del último partido, una comparación de estas tres cifras de rendimiento le permite al programa predecir el resultado probable de un partido determinado.

Si el líder juega en casa contra el último de la clasificación, el resultado puede ser incuestionable, pero el punto fuerte del programa está en predecir el desenlace de un encuentro entre equipos más nivelados. Ello no equivale a afirmar que este sistema acierte con seguridad. El análisis estadístico sugiere que con el programa del profesor George aproximadamente se triplican las probabilidades de éxito: "Admito que los casos no favorables de acierto siguen siendo enormes, pero ¿es o no es mejor apostar con el mayor conocimiento de causa posible?", pregunta el profesor.

La noticia ha saltado a las calles de algunas ciudades españolas, respondiendo a esta pregunta. Varios acertantes de quinielas millonarias lo han sido gracias —según sus propias declaraciones— al ordenador. Parece, pues, que la informática comienza a domesticar el azar.

Las carreras de caballos parecen ofrecer, en todo caso, una mayor esfera de acción para el programador. Un colegial de Darlington ha creado un programa para ordenadores personales que predice los ganadores. Escrito originalmente para el Sinclair ZX-81, y ahora adaptado para el Spectrum, el programa de David Stewart ha dado en el clavo en cierto número de ocasiones.

Quizá sea significativo que los profesionales de las carreras se hayan negado mayoritariamente a utilizar ordenadores. El Jockey Club todavía sigue efectuando manualmente el handicap oficial (aunque los datos se almacenan en ordenador). *Tuneform*, la biblia del aficionado a las carreras, también compila de forma manual la mayoría de sus datos. "Nosotros sólo empleamos el ordenador para calcular las cifras de tiempo promedio para cada caballo, teniendo en cuenta la dirección del viento", explica el director gerente de la publicación, Reginald Griffin. "No existe algo así como un auténtico sistema de handicap informatizado en ningún sitio. El problema es, simplemente, que los ordenadores no pueden dar cuenta de los extraordinarios resultados que suceden a diario."



La informática en las quinielas
Existen varios paquetes para ordenadores personales de los que se afirma que mejoran las probabilidades de ganar en las quinielas, y muchísimos programadores han intentado escribir sus propios programas. Los mejores utilizan una amplia base de datos conteniendo información sobre partidos anteriores, y pueden resultar valiosos para predecir el resultado de partidos decisivos. Como toda forma de apuesta asistida por ordenador, los programas sólo pueden mejorar sus pronósticos de forma marginal, y, por tanto, los proveedores declinan toda responsabilidad



David W. Hamilton

La rueda de la fortuna
En la ruleta, es el número cero el que le proporciona al casino el margen de beneficios. No se puede hacer nada por aumentar las probabilidades del jugador, así que los programas que se desarrollen para este juego se deben concentrar en los sistemas de apuesta

Los ordenadores se están utilizando cada vez más al otro lado del mostrador del establecimiento de apuestas, aunque no para calcular las probabilidades. El personal empleado en las grandes cadenas de corredores de apuestas está entrenado para utilizar calculadoras especiales exclusivamente para averiguar los beneficios sobre las cantidades apostadas. El aspecto crediticio del negocio también se está informatizando cada vez más. Una persona que posea una cuenta en una de las agencias de apuestas pertenecientes a la cadena puede, simplemente, hacer su apuesta comunicándola por teléfono al centro de ordenadores de la cadena. Los detalles se digitan y se adeuda en la cuenta el valor de la apuesta. Si el caballo escogido sale ganador, se calculan las ganancias y se ingresan en la cuenta del cliente.

La mayor parte de los corredores de apuestas se muestran escépticos respecto al uso del ordenador. "Nunca nadie ha propuesto uno que gane constantemente; de lo contrario, no estaríamos aquí", ha declarado el portavoz de la William Hill. Así y todo, fue esta firma la que montó una de las simulaciones por ordenador más extraordinarias y controvertidas de todos los tiempos. Los detalles relativos al estado de los antiguos ganadores del clásico Derby se incorporaron a un programa encargado especialmente. Entonces se invitó a los lectores del periódico a que predijeran los primeros seis. La controversia surgió al asignar un puesto al gran caballo italiano *Ribop*, que jamás perdió una carrera. ¡El ordenador lo colocó en cuarto lugar!

De todos los ordenadores "para apuestas", quizá el más famoso sea ERNIE (*Electronic Random Number Indicator Equipment*: equipo electrónico indicador de números al azar), la máquina que elige los números ganadores de los bonos emitidos

por las cajas de ahorros británicas. Se puede discutir si ERNIE es realmente un ordenador en toda regla; pero, aunque no es programable, sí ejecuta un programa. La máquina la desarrolló Plessey en 1973 para sustituir a la máquina original, construida en 1957. Su función consiste en generar al azar alrededor de 200 000 números y escribirlos en cinta magnética. Estos números se generan a partir de una serie que empieza por el bono premiado que posea el número de emisión más bajo, y termina por el más alto. La cinta se carga luego en un ordenador de unidad principal ICL, que compara los números con una cinta en la que están listados todos los números de aquellos bonos que ya se han reintegrado. Una vez que se han eliminado estos números no elegibles, el ordenador puede imprimir los impresos de cobro para los ganadores.

Desde que se encargaron, los dos ERNIE han generado los números para 22,2 millones de premios, que suman unos 265 millones de pesetas. ¿Le parece mucho? En realidad, no es tanto; las posibilidades de que un bono obtenga un premio en la tirada mensual son de apenas una entre 15 000.

Otro juego de azar muy conocido es el que aparece diariamente en algunos periódicos británicos. Las probabilidades de ganar uno de los premios de un millón de libras que se ofrecen en estas campañas de promoción son todavía más remotas. Los números de los cupones que se distribuyen a los lectores tienen una longitud de 12 dígitos. Una secuencia de dígitos comprendida entre 000000000000 y 999999999999 ofrece un billón de combinaciones distintas. Estadísticamente, las probabilidades en contra de que una mañana determinada aparezca un número dado serían algo así como un millón de millones contra uno. Sobre esta base, resulta sumamente dudoso que el periódico tenga que pagar *alguna vez* el premio gordo.

La situación se puede visualizar mejor si nos imaginamos un bolso que contenga dos millones y medio de bolas blancas, que representan a los participantes (el número de cupones en circulación), y un billón de bolas negras que representan el total de números distintos posibles. Huelga decir que la probabilidad de extraer una bola blanca a la primera es más bien remota. Además, la probabilidad no mejora significativamente ni siquiera después de que se hayan extraído las correspondientes a un año. Las probabilidades de que el periódico tenga que pagar un millón de libras son de una entre 667.



Un golpe de suerte
En *blackjack* (o "veintiuno") existe como juego para la mayoría de los ordenadores personales, y proporciona una de las mejores pautas para escribir programas ganadores. La capacidad de memorizar los naipes que ya se han jugado aumenta las probabilidades de éxito del jugador, aunque como los casinos no admiten ordenadores en las mesas de juego, en todas las hazañas célebres han participado ordenadores ocultos (ha habido un caso en que fue atado a la pierna del jugador, cubierto con la pernera de su pantalón) o enlaces por radio con máquinas situadas en el exterior

Cargando los dados

El ingrediente básico de las apuestas, la generación de números al azar, se puede simular fácilmente en un ordenador personal. La mayoría de las versiones de BASIC proporcionan una función generadora de números aleatorios. Sin embargo, en muchos casos los números que se generan de este modo no son auténticamente al azar, como demuestra el breve programa que ofrecemos a continuación:

```
10 LET A = RND
20 LET B = RND
30 LET C = RND
40 PRINT A,B,C
```

En cada una de las tres primeras líneas, se les asigna un número supuestamente al azar a las variables A, B y C. Luego, éstas se imprimen. Podremos obtener los resultados siguientes (aunque los suyos diferirán):

```
.014007 .964370 .457397
```

Pero si vuelve a ejecutar el programa, la mayoría de los microordenadores visualizará otra vez la misma secuencia. Y se explica: cuando solicitamos RND, el ordenador responde con el siguiente número de una secuencia fija. Por norma, ésta puede abarcar el millón de fracciones de seis dígitos entre 0,000000 y 0,999999, cada una de las cuales se selecciona una vez en el ciclo completo (pero no en secuencia).

Ciertas versiones de BASIC utilizan una sintaxis ligeramente diferente, que exige una expresión entre paréntesis denominada *argumento*. Éste asume la forma LET A = RND(X). El efecto es muy similar: tanto RND como RND(X) se pueden utilizar de la misma forma que otras variables.

Algunas versiones de BASIC también incorporan una función RANDOMIZE, que hace que la secuencia empiece en un punto impredecible. Insertando la orden RANDOMIZE en las primeras líneas de cualquier programa en el cual se utilice RND, se asegura que cada vez que se ejecute el programa se genere una secuencia de números distinta.

Para simular el lanzamiento de un dado necesitamos enteros en la escala entre 1 y 6. O sea, es necesario eliminar las fracciones. Esto se realiza utilizando la función INT (entero). PRINT INT(6.99) produce como resultado 6, al igual que PRINT INT(6.01) producirá también 6. Todo cuanto venga después de la coma decimal se descarta.

Puesto que el máximo número que puede generar RND es .999999, se requiere una pequeña multiplicación. La fórmula generalmente aceptada es:

```
LET A = INT(6*RND) + 1
```

Multiplicamos por seis porque un dado posee seis caras. El "más uno" es sencillamente para asegurar que el resultado esté comprendido entre 1 y 6, y no entre 0 y 5.

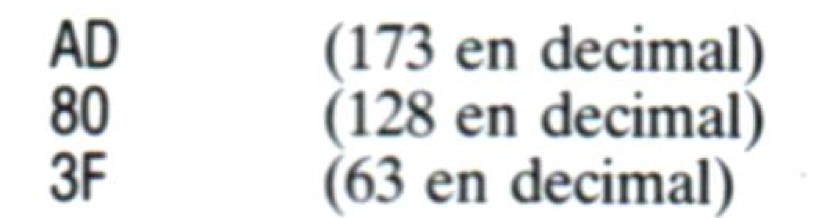
El lenguaje ensamblador

Continuando con nuestra introducción al código de lenguaje máquina, analizamos las formas de expresar los programas, desde los números binarios al lenguaje ensamblador

Una de las dificultades conceptuales que experimenta la mayoría de los recién iniciados en el código de lenguaje máquina es el hecho de que los programas puedan asumir diversas formas. Todo dato almacenado en la memoria del ordenador, en última instancia asume la forma de números binarios de ocho dígitos. Sin embargo, cuando éstos se escriben sobre papel, ocupan muchísimo espacio, son difíciles de leer y recordar y propician los errores de digitación. Por eso se recurre a los números en base dieciséis (hexa), pues ofrecen las ventajas de que el contenido de un byte se puede expresar como un número de dos dígitos y que cualquier dirección en la memoria del ordenador (de 0 a 65535 en decimal) se puede representar mediante cuatro dígitos.

Cuando escribimos un número hexa sobre el papel por lo general le ponemos delante un signo $ para distinguirlo de los números decimales, bien sabiendo que cuando se da entrada al programa el signo no se incorpora a la memoria del ordenador. Además, cuando un opcode posee un operando de dos bytes (p. ej., LDA $3F80), a los dos bytes se les da entrada en la máquina por orden inverso; es decir, el byte de la derecha seguido del byte de la izquierda. En el ejemplo anterior, por tanto, los tres bytes serían AD (la representación hexa del opcode LDA en lenguaje 6502) seguido de 80, seguido de 3F. Ello simplifica mucho las cosas para el procesador, pero puede resultarle confuso al usuario.

Normalmente, un programa en código de lenguaje máquina se imprime como un *dump* (volcado) en hexadecimal: una larga lista de valores hexadecimales de dos dígitos. Además, se dará una dirección de comienzo (ya sea en hexa o en decimal) y es en esta posición donde se debe cargar el primer valor hexa, el segundo en la posición siguiente, y así sucesivamente. La carga se puede conseguir mediante la orden POKE del BASIC. Si la dirección de comienzo es $1000 (4096, en decimal), y el *dump* hexa es:



AD (173 en decimal)
80 (128 en decimal)
3F (63 en decimal)

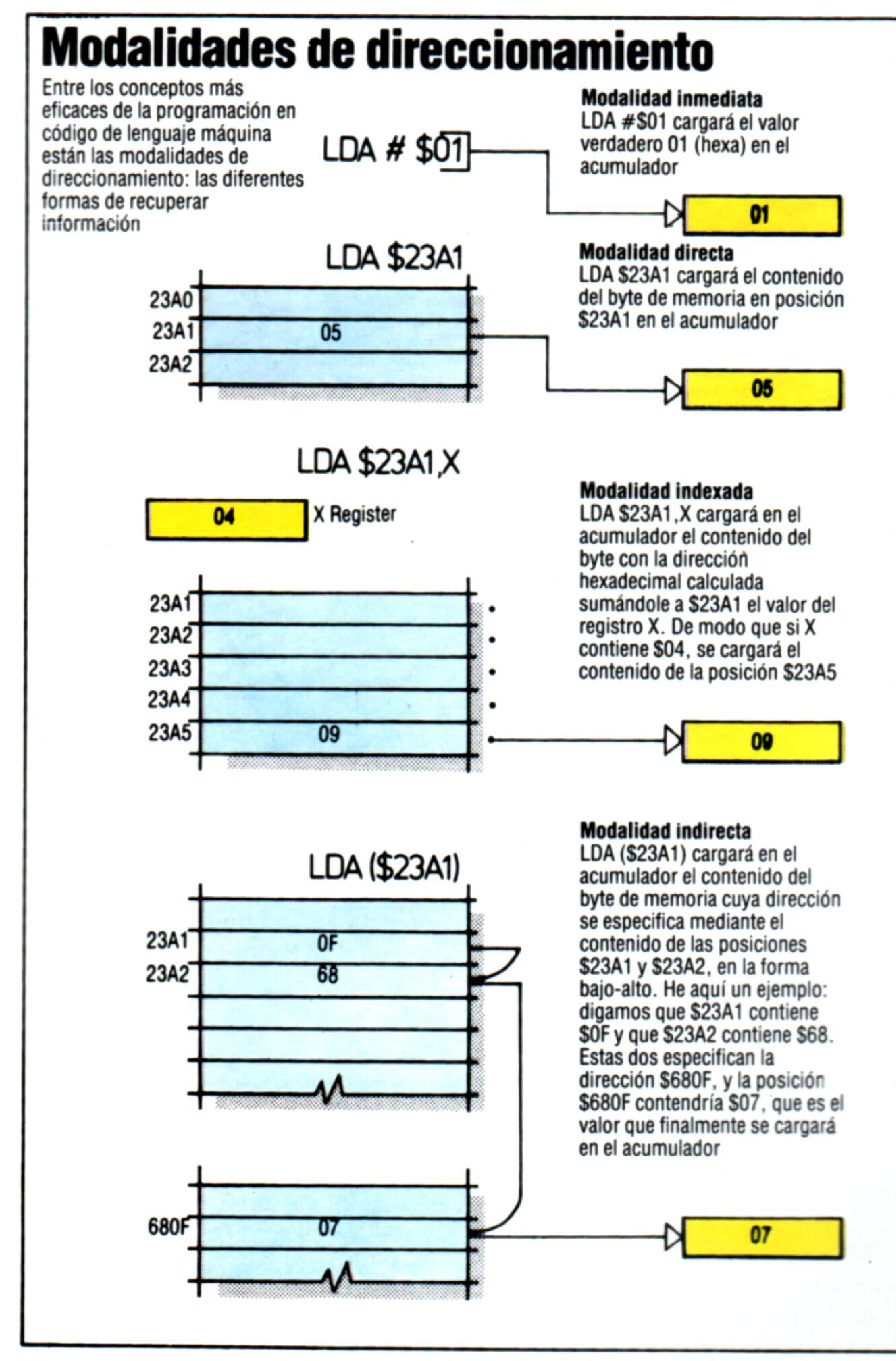
el programa se puede cargar con estas tres sentencias en BASIC:

```
POKE 4096,173
POKE 4097,128
POKE 4098,63
```

Observe cómo hemos de convertir todos los valores de hexa a decimal antes de que los podamos utilizar en la sentencia POKE; en el interior de la máquina se almacenarán en binario.

Para *dumps* hexa más largos es normal emplear un breve programa en BASIC denominado *cargador de código máquina*. Éste pregunta la dirección de comienzo y después los valores hexa. A medida que se les va dando entrada a cada uno, la breve rutina en BASIC convierte el valor hexa en decimal, y lo coloca (POKE) en la posición siguiente. Si se desea, el *dump* hexa lo puede leer (READ) el programa por medio de sentencias DATA.

Una vez que se ha cargado el código máquina, se puede prescindir del programa cargador en BASIC. Por consiguiente, es importante cargar el código máquina en algún lugar de la memoria donde el



Modalidades de direccionamiento

Entre los conceptos más eficaces de la programación en código de lenguaje máquina están las modalidades de direccionamiento: las diferentes formas de recuperar información

Modalidad inmediata
LDA #$01 cargará el valor verdadero 01 (hexa) en el acumulador

Modalidad directa
LDA $23A1 cargará el contenido del byte de memoria en posición $23A1 en el acumulador

Modalidad indexada
LDA $23A1,X cargará en el acumulador el contenido del byte con la dirección hexadecimal calculada sumándole a $23A1 el valor del registro X. De modo que si X contiene $04, se cargará el contenido de la posición $23A5

Modalidad indirecta
LDA ($23A1) cargará en el acumulador el contenido del byte de memoria cuya dirección se especifica mediante el contenido de las posiciones $23A1 y $23A2, en la forma bajo-alto. He aquí un ejemplo: digamos que $23A1 contiene $0F y que $23A2 contiene $68. Estas dos especifican la dirección $680F, y la posición $680F contendría $07, que es el valor que finalmente se cargará en el acumulador

Versión íntegra y abreviada

Un programa en código de lenguaje máquina puede asumir varias formas diferentes. El programador lo suele escribir en forma de lenguaje ensamblador, que utiliza mnemotécnicos para los opcodes y etiquetas para los operandos, de modo que:

```
LDA PESO
ADC COMBUSTIBLE
STA PESO
```

Debemos, no obstante, especificar las direcciones de aquellas etiquetas. Por ejemplo:

```
COMBUSTIBLE = $03EE
PESO = $031F
```

Un paquete ensamblador transformaría esto en un *dump* hexa, utilizando una unidad de disco. El "lenguaje seudoensamblador", como vemos más abajo, resulta menos fácil de leer, pero a menudo se le puede dar entrada en un paquete denominado *ensamblador puntual*, que no requiere discos.

```
LDA $031F
ADC $03EE
STA $031F
```

Un *dump* hexa consta de una dirección de comienzo (a la izquierda) y la secuencia de valores hexa de dos dígitos como aparecerán en la memoria. Observe que un operando como $031F se almacena en orden inverso (1F 03) y que los opcodes se han sustituido por el valor hexa adecuado:

```
19C4 AD 1F 03 6D EE 03 8D 1F 03
```

programa en BASIC no lo "machaque", ni donde sentencias como NEW lo destruyan.

La mayoría de los ordenadores personales posee alguna orden en BASIC para indicarle a la máquina que deje de ejecutar BASIC y que empiece a ejecutar el programa en código máquina a partir de un punto determinado. Una forma de esta orden es SYS 4096 (RETURN), que significa "transfiere el control al sistema comenzando por la posición decimal 4096"; otra forma es CALL $E651, que significa "llama a la rutina en código máquina a partir de la posición hexa E651".

El programa o la subrutina en código máquina ejecutará entonces este sistema o rutina (con los resultados visibles o no, según la naturaleza del programa). Si está escrita correctamente y si incorpora el procedimiento de finalización adecuado, el control se volverá a pasar al BASIC. Entre paréntesis, esto significa que se puede llamar a subrutinas en código máquina desde varios lugares durante la operación de un programa en BASIC cuando sea necesario efectuar una función a gran velocidad.

Uno de los inconvenientes de programar en código máquina es que si usted ha cometido un error en su código, el ordenador no le asistirá con su útil y caritativo mensaje ERROR DE SINTAXIS. En cambio, es más que probable que le "reviente": la máquina no responderá a nada que le digite. Esto no es perjudicial para el ordenador, pero tendrá que restablecerlo (o apagar la máquina y volver a encenderla) y esto por norma significa que tendrá que volver a introducir de nuevo el programa partiendo de cero. Ésta es la razón por la cual uno no puede experimentar en código máquina del modo como puede experimentar en BASIC: la viabilidad de un programa debe verificarse exhaustivamente sobre el papel antes de darle entrada en el ordenador.

No obstante, un dispositivo de software que puede ser de gran ayuda para dar entrada y verificar el código máquina es el *monitor de código de lenguaje máquina* (que no tiene nada que ver con un monitor de pantalla). Algunos ordenadores, muy pocos, lo llevan incorporado en su ROM, pero por lo general se compra en forma de paquete sobre cassette o cartucho. Un monitor de código máquina es un sistema operativo sencillo que visualiza en la pantalla el contenido de cualquier sector de la memoria que se solicite. Estos valores (hexa) se pueden simplemente modificar o bien volver a escribir sobre ellos, de modo que un monitor es, con mucho, la forma más rápida de dar entrada a un *dump* hexa. Además, por lo general puede cargar y guardar los programas en código máquina directamente en cassette, sin necesidad del programa cargador en BASIC. Los más avanzados programas de utilidades en código máquina (el equivalente en código máquina a los equipos de herramientas de BASIC; véase p. 444) muestran el contenido de cada uno de los registros internos del procesador.

Los *dumps* hexa constituyen una conveniente forma de expresar el código de lenguaje máquina, pero no son fáciles de leer. A menos que recuerde el equivalente hexadecimal de todos los diversos opcodes, es casi imposible distinguirlos de los operandos. De modo que los programas se suelen escribir con un artificio mnemotécnico de tres letras que ya vimos en el capítulo anterior (véase p. 449), y éstas se traducen a hexa con una tabla de códigos que figura en el manual del microprocesador.

Sin embargo, un tipo más refinado de monitor de código máquina le permitirá digitar el programa mnemotécnico en su totalidad y obtener las conversiones de manera automática. Es el llamado *ensamblador puntual*, porque ensambla las letras mnemotécnicas en forma de números sobre la marcha.

Esto nos lleva a la forma final en la que se puede expresar el código máquina, el lenguaje ensamblador, que no sólo recurre a la mnemotecnia para los opcodes sino que puede manipular nombres (o etiquetas) en vez de los números hexas para los operandos. Por tanto, si la posición $07B2 contiene el número actualizado de misiles que se disparan en un juego, nosotros podemos cargarlo en el acumulador con la instrucción:

```
LDA MISSIL
```

Al comienzo del programa tendremos que especificar la posición MISSIL = $07B2, y asegurar que contenga inicialmente el valor $09 (nueve misiles).

Cuando hemos acabado de desarrollar este programa en lenguaje ensamblador (denominado *código fuente* del programa), ejecutamos un programa de utilidades que recibe el nombre de *ensamblador*. Éste revisa el código, sustituyendo las palabras mnemotécnicas y todas las etiquetas por sus equivalentes hexas y creando, por consiguiente, una nueva versión denominada *código objeto*. A este código se le puede dar entrada en la memoria del ordenador y se puede ejecutar. El proceso es similar al de la compilación (véase p. 84), si bien en este caso existe una correspondencia individualizada entre el código fuente y el código objeto.

El lenguaje ensamblador, al ser un lenguaje de mayor nivel que el código de lenguaje máquina, es considerablemente más fácil de escribir, sin merma de eficacia. No obstante, los paquetes ensambladores por lo general sólo funcionan con una unidad de disco, y por ello no están al alcance de todos los usuarios de ordenadores personales.

Opcodes

He aquí algunos opcodes más, que un microprocesador corriente incorporará

JSR

Jump SubRoutine
(Salto a subrutina)
Esta función equivale a la GOSUB del BASIC. JSR $354D modificará el contenido del registro contador (PC) del programa de modo que ejecute el código desde $354D hacia adelante

RTS

ReTurn from Subroutine
(Regreso de subrutina)
Al encontrarse con RTS, el procesador saltará hacia atrás hasta la posición desde la cual se llamó a la subrutina (o sea, equivale a RETURN en BASIC). RTS no tiene operando, porque la dirección de retorno se habrá almacenado automáticamente en una zona especial de la memoria denominada *stack* (pila, montón)

BMI

Branch if MInus
(Bifurcación si menos)
Ésta es una de las varias formas de bifurcación condicionada en código máquina (en BASIC, IF...THEN GOTO es una bifurcación condicionada). Si en la última operación se obtuvo como resultado un valor negativo en el acumulador, la ejecución del programa saltará a una dirección especificada. BPL hace lo mismo cuando se obtiene un signo positivo (PL*us*)

LDX

LoaD X register
(Carga en registro X)
X es un registro de un único byte dentro del microprocesador, y si bien no puede efectuar aritmética del mismo modo que el acumulador, se lo utiliza para el "direccionamiento indexado" (véase cuadro). LDX carga un valor en X, y STX (STore X: almacenar X) lo volverá a almacenar en la memoria

INX

INcrement X
(Incrementar X)
Sumando 1 al valor de X (DEX le restaría 1: DEcrement X) y utilizando el direccionamiento indexado, se puede atravesar un número de posiciones de la memoria, realizando en cada una de ellas el mismo proceso

El futuro

Los últimos cinco años han resultado cruciales para la revolución informática; pero ¿qué nos deparará el próximo lustro?

¿Qué aspecto tendrá el ordenador personal de la última década del siglo xx y cómo funcionará? Son preguntas a las que tratamos de dar respuesta en este artículo, repasando uno a uno los principales componentes y sistemas de la máquina del mañana. Muchas de estas ideas se basan en tecnologías que están comenzando a salir al mercado (quizá en otros campos distintos de la informática), mientras que otras representan lo que nosotros creemos que son desarrollos probables.

Una de las configuraciones más fundamentales de nuestro diseño hipotético es la modularidad. Habiendo adquirido la unidad base, el usuario dispondrá de una amplia gama de opciones para ampliar la máquina. En realidad, el usuario podrá virtualmente diseñar su propia máquina seleccionando este módulo para gráficos y aquella facilidad para el sonido. Algo es seguro: la velocidad de los cambios en el mercado del ordenador seguirá en continua aceleración muchos años más.



1 Visualización de teclado

La potencia del microprocesador de 32 bits permitirá visualizar la información en varias formas simultáneamente. Por ejemplo, la pantalla principal podría mostrar la vista que se contempla desde el sillón de mando de una nave espacial, mientras una pantalla subsidiaria montada sobre la consola de mando-teclado podría visualizar la información de control desde la cabina

2 Teclado

A pesar de la innata ineficacia del teclado QWERTY, es poco probable que se efectúen intentos serios por establecer un modelo alternativo. Las teclas con resorte estilo máquina de escribir son con mucho las que gozan de mayor popularidad, aunque es probable que se extienda la utilización de las teclas de efecto Hall, que emplean imanes en vez de resortes. Los interruptores electrónicos propiamente dichos podrían reemplazarse por un sistema que se basa en que las teclas conmutan una matriz de rayos láser

3 Monitor

Los televisores de proyector existen desde comienzos de la década de los 80, pero su campo de acción se ve limitado por la potencia de emisión de luz del tubo de rayos catódicos (TRC). Es probable que la tecnología TRC ponga a nuestro alcance sistemas de proyección que abarquen todo el ancho de una habitación. Los primeros televisores de proyector necesitaban pantallas curvadas especiales, pero los modelos más recientes ya se enfocan contra una superficie plana



Tony Lodge

4 Procesadores alternativos

Además del procesador principal de 32 bits, es probable que el micro dentro de unos años albergue procesadores adicionales a manera de módulos enchufables. El procesador principal podría entonces "delegar" parte del procesamiento (p. ej., la operación de un periférico determinado o clasificar un archivo de datos) al subprocesador más idóneo. Además, los módulos enchufables baratos podrían emular a los ordenadores clásicos de los años ochenta, de modo que el software de cualquier otro ordenador se podría ejecutar sin ninguna modificación

5 Memoria de acceso directo

El procesador de 32 bits puede direccionar hasta casi 4 300 millones de posiciones de memoria, lo que representa una diferencia abismal en relación al límite de 65 536 bytes impuesto por los procesadores de ocho bits con los que han entrado los microordenadores en el hogar

6 Comunicaciones

Mientras que las antenas parabólicas para recibir señales emitidas por satélites serán algo corriente en los años noventa, y la mayoría de los canales telefónicos estarán digitalizados en vez de basarse en señales analógicas, aun entonces será necesario regular la velocidad de transmisión y recepción. Estos controladores de comunicaciones realizarán parte de las funciones de control de los moduladores-demoduladores actuales

7 Alimentación eléctrica

La creciente carga y multiplicidad de dispositivos conectados al microordenador probablemente exijan una fuente de alimentación eléctrica significativamente mayor que las que se utilizan en la actualidad. Incorporará circuitos estabilizadores y el apoyo de pilas de reserva recargables, de modo que las fluctuaciones o los cortes de potencia de la red eléctrica no ocasionen la pérdida o el deterioro de los datos

8 Pantalla portátil

La tecnología de pantalla plana (que probablemente utilice una matriz de cristal líquido quizá conectada al procesador central mediante una conexión de infrarrojos o incluso de microondas) se podría emplear para visualizar texto y material gráfico. Si este dispositivo fuera, además, sensible al tacto, podría ser a la vez tablero para selección de menús y digitalizador

9 CDROM

La *Compact Disk ROM* (ROM en disco compacto), que utiliza un rayo láser para leer información codificada ópticamente, tiene grandes posibilidades de reemplazar a los cartuchos de ROM convencionales; debido a su capacidad: una CDROM típica retendría cuatro megabytes

10 Diskettes flexibles

Para finales de la década, los diskettes flexibles habrían de alcanzar un desarrollo como para competir con los discos Winchester, tanto en velocidad como en densidades de almacenamiento de datos. Al mismo tiempo, su diámetro se habría de reducir a menos del mínimo corriente de 3 pulgadas

11 Panel frontal

Antes del advenimiento de los lenguajes de alto nivel y de los teclados, se había de dar entrada a los programas en notación binaria mediante el panel frontal: una línea de luces e interruptores que le proporcionaban al usuario el control sobre cada bit de los buses de direcciones, de datos y de control. Para los entusiastas del código de lenguaje máquina, el panel frontal aún puede ser una herramienta útil, de modo que esta idea podría resurgir en los ordenadores personales futuros

12 Ratones infrarrojos

El IBM PC-Junior ya utiliza la radiación infrarroja para transferir los datos del teclado al ordenador sin ninguna conexión por cable. Esta tecnología podría proporcionar la interconexión entre todos los periféricos, incluyendo los ratones, eliminando por consiguiente el "efecto spaghetti". Existirán, por supuesto, modelos tanto para diestros como para zurdos

13 Microprocesadores

Los primeros ordenadores personales basados en microprocesadores de 32 bits salieron al mercado en 1983, pero no tuvieron más remedio que depender de buses de datos de 16 bits, e incluso de 8, para mantener la compatibilidad con los chips de memoria y los periféricos existentes, y no dieron el potencial que prometían. Con la introducción de dispositivos como el chip Motorola 68032, que ofrece un procesamiento de 32 bits y una transferencia de datos de 32 bits, las capacidades de estos procesadores de gran capacidad se generalizarán. Muchos miniordenadores de alto precio poseen ya procesadores de 32 bits

Pugna generacional

Con la introducción de la tecnología VLSI, estamos actualmente a punto de entrar en la cuarta generación de ordenadores. Pero los japoneses anuncian ya una quinta generación

Se dice que los ancianos no hacen revoluciones, y el director del proyecto japonés para la creación de la quinta generación de ordenadores parece haberse tomado al pie de la letra estas palabras. Al seleccionar 40 científicos de las diez mayores corporaciones y de los laboratorios del gobierno para que trabajaran con él en el Instituto para la Tecnología de la Nueva Generación de Ordenadores, en Tokio, el doctor Kazuhiro Fuchi los ha escogido entre aquellos que tenían menos de 35 años de edad. El Instituto se fundó el 14 de abril de 1982 con un presupuesto de alrededor de 70 000 millones de pesetas (a gastar en diez años) y es una empresa de cooperación entre el gobierno y la industria. Empresas como Fujitsu, Sharp y Toshiba participan en este ambicioso proyecto, que tiene como objeto dar un salto respecto al estado actual de la tecnología de ordenadores.

La acuñación del término *quinta generación* centra la atención en los principales avances del pasado en el campo del diseño de ordenadores, estimulando fantásticas posibilidades para el futuro. La primera generación de ordenadores se caracterizó por la utilización de válvulas termoiónicas, pronto envejecidas con la invención del transistor. Los ordenadores de transistores de la segunda generación fueron, a su vez, superados por las máquinas que utilizan la tecnología de integración a gran escala (LSI: *Large Scale Integration*), consistente en incorporar muchos transistores en un solo chip. Ahora mismo estamos ante el final de esta tercera generación de ordenadores, pero los últimos años de la década de los ochenta deberían ver la aparición en el mercado de los chips VLSI (*Very Large Scale Integration*: integración a escala muy grande) de la cuarta generación. Estos chips tendrán hasta diez millones de transistores por chip, frente al límite actual de aproximadamente un cuarto de millón.

En la actualidad, la IBM invierte al año más de 230 billones de pesetas en investigación y desarrollo de ordenadores, cifra que convierte a nuestros ojos la inversión japonesa en algo insignificante. Pero el desembolso del capital japonés no obedece sólo a razones estrictamente lucrativas.

La ciencia ha desplazado durante los últimos cien años su foco de interés desde el aprovechamiento de la energía en bruto (amplio abanico que va de la electricidad al motor de combustión interna) hasta el estudio de la forma de riqueza más intangible: la información. La tierra, el trabajo, el capital y la industria pueden haber sido en el pasado las fuentes de poder, pero el futuro favorecerá a aquéllas relativas al control de la información. El conocimiento y el procesamiento de la información serán las claves de la sociedad postindustrial. De modo que lo que se necesita para esta nueva sociedad es un ingenio, una máquina con razonamiento automático que se pueda aplicar a cualquier problema concreto o campo del esfuerzo humano con la precisión matemática y la certeza típicas de un ordenador. El ingenio que están construyendo actualmente los japoneses se denomina KIPS (*Knowledge and Information Processing System*: sistema de procesamiento de la información y el conocimiento).

Los seres humanos son muy buenos en cuanto a convertir señales sensoriales en formas cognoscitivas (la situación en una partida de ajedrez se puede colegir con una mirada), pero cuando se trata de tomar decisiones que dependen de grandes cantidades de datos, nuestras limitaciones quedan enseguida de manifiesto. Las reglas del ajedrez se pueden explicar en unos pocos minutos, y, sin embargo, el juego es tan complicado que los grandes maestros sólo son conscientes de una docena de posibles movimientos. No obstante, en principio todo problema al cual se le aplica el raciocinio se puede dividir en una serie de pasos simples, cada uno de los cuales se puede decidir mediante la aplicación de reglas deductivas. Este conjunto de reglas se conoce como lógica de predicados. Las reglas deductivas de la lógica se aplican a todos los problemas, pero cuando tomamos decisiones cotidianas sencillas no somos conscientes de ellas.

Lenguaje lógico
El PROLOG (abreviatura de *Programming Logic*: lógica de programación) se desarrolló a principios de los años 70 en el seminario de inteligencia artificial de la Universidad de Marsella, si bien uno de los principales protagonistas de su desarrollo y promoción es el norteamericano Robert Kowalski, del Imperial College de Londres. Basado en algunos de los principios de la lógica humana, es el lenguaje que tiene más probabilidades de utilizarse en los ordenadores de la quinta generación. Es también muy útil para crear y tratar bases de datos y para fines educativos

Un experto necesita algo más que un buen cerebro. Pongamos por caso un doctor, que necesita muchos años de entrenamiento para acumular su conocimiento médico. De la misma manera, un KIPS debe poseer un banco de datos sobre el cual puedan operar las reglas de inferencia. Además, el sistema ha de ser sumamente amable con el usuario si se desea que el KIPS no exija su propia cantera de expertos para operarlo. Una máquina KIPS con la cual uno pueda mantener una conversación en el lenguaje que se elija debe ser producto de la investigación en el campo de la inteligencia artificial, que es un área de estudio muy controvertida. Por consiguiente, los objetivos que los japoneses se han fijado a sí mismos abarcan una amplia gama de la ciencia del ordenador: hardware, software, interfaces, sistemas especializados (véase p. 72) y los problemas de la inteligencia artificial.

El proyecto japonés se ha concebido para ir más allá de los adelantos en la tecnología del chip. A medida que aumenta la densidad de transistores de

los circuitos integrados, menor es la distancia que han de recorrer los electrones entre un componente y otro y, en consecuencia, los circuitos operarán con mayor rapidez. Pero no es la velocidad en sí lo que importa a los japoneses, y por eso sus esfuerzos se centran en el software. En una partida de ajedrez, por ejemplo, son tantas las posibles secuencias de movimientos (alrededor de 10^{120}) que se calcula que el tiempo necesario a la mente humana para explorar todas las posibilidades se aproxima al que le queda de vida a nuestro Sol. Uno de los objetivos del proyecto es la producción de la máquina que pueda realizar 100 millones de inferencias lógicas (es decir, que pueda aplicar 100 millones de reglas) por segundo. En este sentido se dice 100 millones de LIPS (*Logical Inferences Per Second*: inferencias lógicas por segundo).

Otra manera de aumentar la velocidad sería cincelando las funciones de software en el diseño del chip, en vez de cargarlas en la memoria y procesarlas por medio de un chip de finalidad universal. Este intento de superar la distinción entre hardware y software es una de las metas más interesantes del proyecto. Ya existen memorias "asociativas" que tienen circuitos de búsqueda lógica incorporados en las celdas de memoria. Estos dispositivos pueden posicionar un dato exclusivamente a partir del significado del propio dato, sin necesidad de especificar una dirección de memoria.

Los avances de este tipo producirán un interesante maridaje entre los procesadores lógicos y los bancos de datos. Cincelar circuitos con rutinas de programación en un procesador recuerda los primeros ordenadores como el ENIAC (véase p. 140), pero las máquinas de la quinta generación se diferenciarán de la arquitectura de Von Neumann en un aspecto fundamental: incorporarán muchos procesadores distintos que funcionarán todos simultáneamente (en paralelo), en lugar de una sola unidad central de proceso. Esto exigirá especial esmero en cuanto a la sincronización y el control de las operaciones internas, pero eliminará esa merma de velocidad que impone la ejecución secuencial de las instrucciones. El lenguaje interno escogido para el KIPS es el PROLOG, un lenguaje que se desarrolló en Francia y Gran Bretaña y que se basa en la lógica de predicados. Pero el KIPS tendrá la capacidad de comunicarse con sus usuarios en muchas lenguas.

La traducción del habla natural humana es otro de los objetivos del proyecto, con el directo propósito de un 95 % de precisión. La capacidad para reconocer palabras individuales pronunciadas por diferentes interlocutores va muy a la zaga comparada con el éxito de la voz sintética. No obstante, la NEC Corporation, de Japón, ya ha creado una máquina capaz de reconocer el habla normal.

Con respecto a la palabra escrita, el proyecto tiene en preparación un diccionario y un programa japonés-inglés de 100 000 palabras, y se espera que permita traducir con una precisión del 90 %.

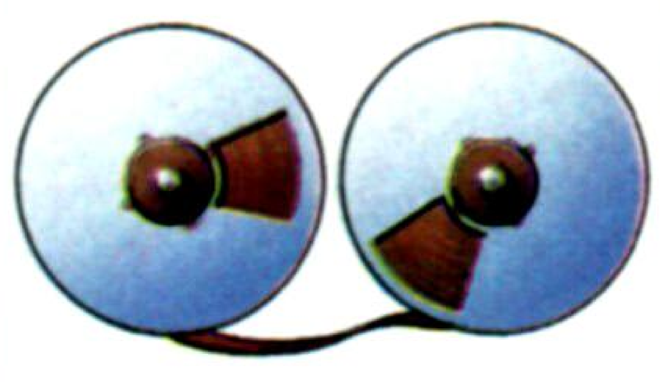
Japón ya tiene precedentes en cuanto al éxito de proyectos de investigación a largo plazo: el proyecto PIPS (*Pattern Information Processing Systems*) de la década de los setenta está resultando muy útil para el desarrollo de bancos de datos visuales e interfaces de fácil manejo para el usuario. Un KIPS podrá ser capaz de mirar una imagen y extraer sus contornos y sus características más destacadas, para hacerse una idea preliminar. En el metro de Tokio ya existe una máquina que explora los corredores con una cámara de video y presenta un cuadro del flujo de pasajeros en las diferentes líneas.

En Estados Unidos, la tecnología de la información representó en 1983 un movimiento por valor de 88 billones de dólares. Con una perspectiva decreciente del nivel de empleo en la industria fabril semejante al de la agricultura a principios de este siglo (del 40 % de la mano de obra empleada en agricultura al comienzo del siglo se pasó al 3 % en la actualidad), la sociedad se ha visto abocada cada vez más a una cultura de la información. Bajo este prisma, Japón está intentando algo muy ambicioso con su proyecto para la quinta generación. El plan es optimista y discurre por unos derroteros "preseleccionados" que tendrán o no viabilidad (al fin y al cabo, todavía esperamos transitar por la tan anhelada senda de la fusión nuclear controlada). Pero se trata de un enfoque positivo por parte de un país eminentemente exportador, muy similar a cualquier estado occidental avanzado. Sin embargo, mientras que, por ejemplo, en Gran Bretaña, la inversión en investigación ha disminuido en la última década, Japón sigue especulando con su futuro.

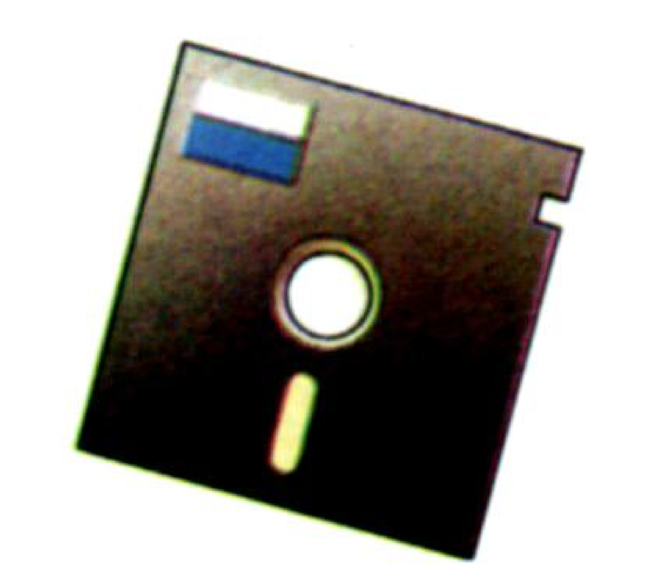
Generaciones en liza



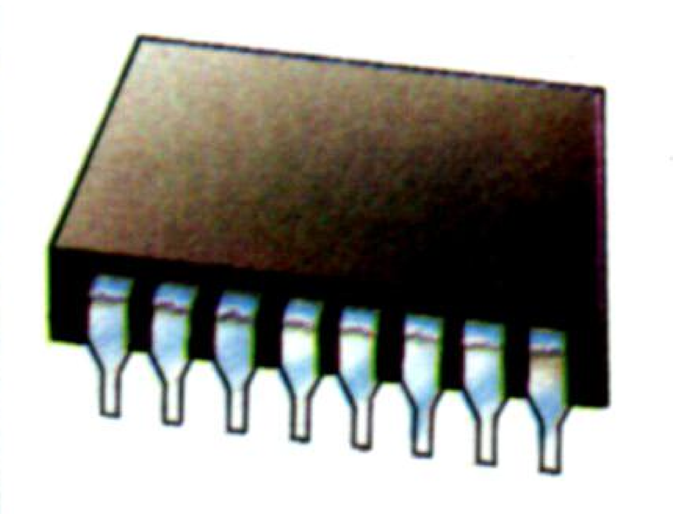
Primer asalto
La primera generación de ordenadores electrónicos se desarrolló alrededor de la tecnología de la válvula termoiónica. Con muy poca memoria interna, los datos generalmente se almacenaban en tarjetas perforadas



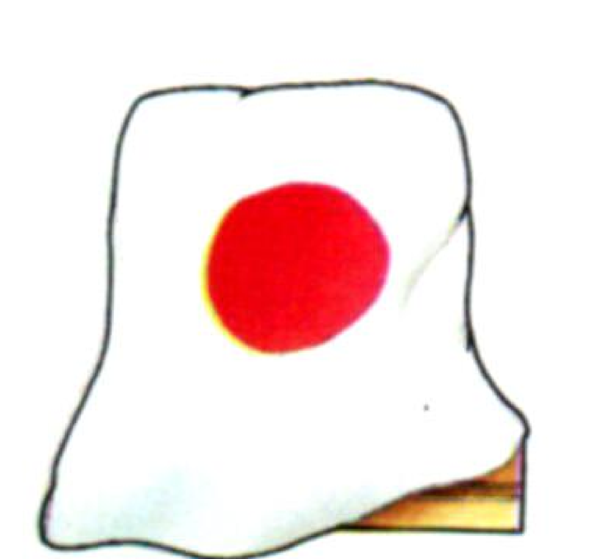
Segundo asalto
La segunda generación se desarrolló a partir del transistor, que aumentó la capacidad de memoria, aunque aún se utilizaba el almacenamiento externo (en forma de cinta magnética)



Tercer asalto
La invención del circuito integrado incrementó sobremanera la potencia del ordenador y permitió el nacimiento del microordenador, caracterizado por la unidad de disco flexible



Cuarto asalto
Ahora estamos pasando de la tercera generación a la cuarta, que se basará en la tecnología del chip VLSI. La memoria RAM será tan grande que el almacenamiento externo tendrá cada vez menos importancia



Quinto asalto
La quinta generación de ordenadores, que se está desarrollando fundamentalmente en Japón, en realidad atañe al software y no al hardware. No obstante, se basa en el supuesto de que la memoria para el usuario será tan grande que el tamaño del programa dejará de ser un factor digno de consideración

Kevin Jones

Research Machines 380Z

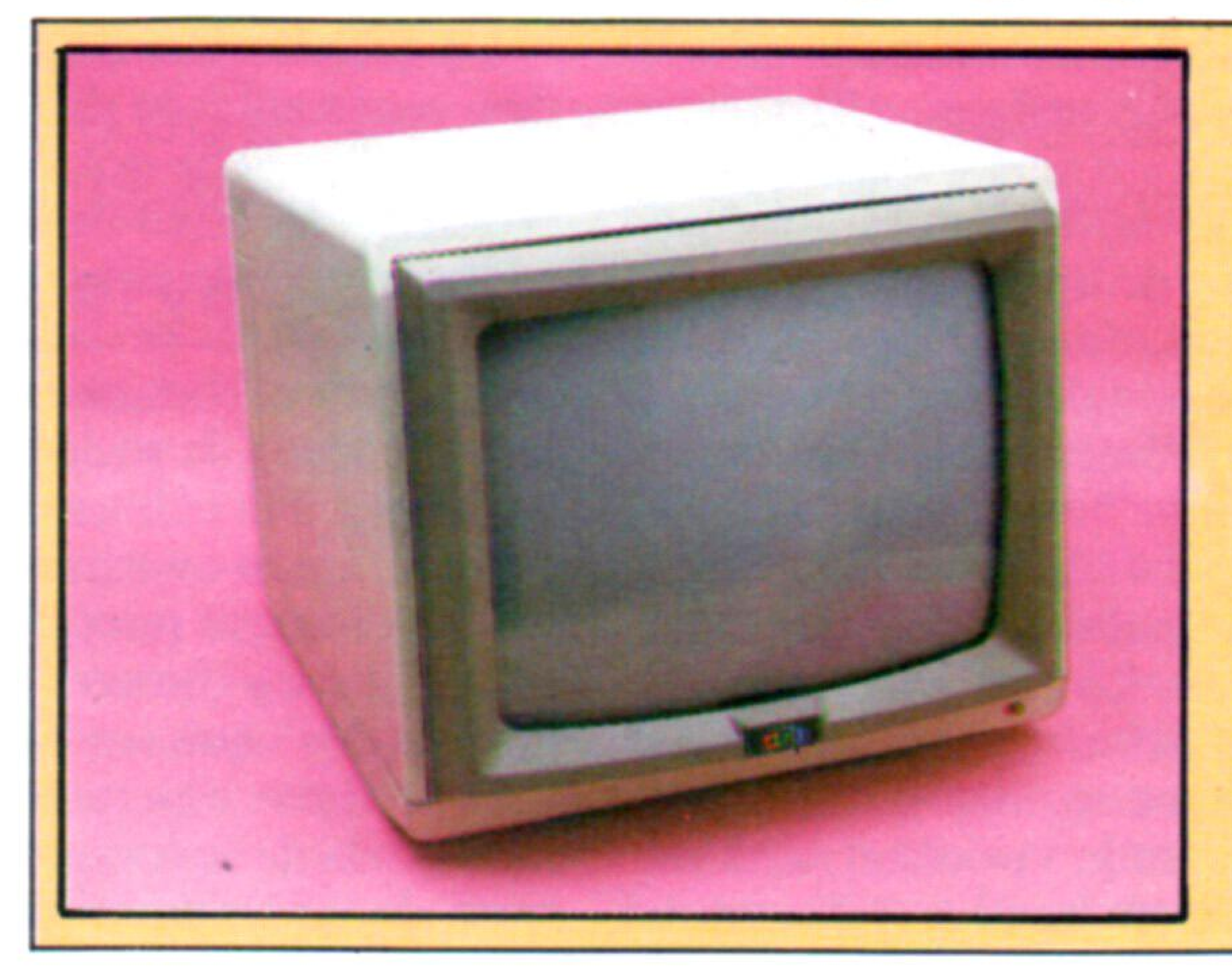
Un sistema físico sólido y unos soberbios gráficos en alta resolución han popularizado este microordenador en las escuelas y entre los militares

Los productos de Research Machines Limited se encuentran entre los modelos de ordenadores más duraderos que existen. Aunque estas máquinas no son especialmente innovadoras ni su precio es competitivo, están diseñadas y construidas con suma solidez, están bien respaldadas y son extraordinariamente fiables. El ordenador más popular de la empresa, el RML 380Z, puede que falte en muchos hogares, pero al ser una de las máquinas más solicitadas por los pedagogos, muchos niños realizaron con él sus primeras experiencias de informática.

Comparado con la mayoría de las máquinas, el 380Z es enorme: los tableros del circuito principal están alojados en una sólida carcasa de 48 cm de ancho, con asas a los lados. Al quitar la tapa de esta "caja negra" descubrimos por qué es tan grande, ya que alrededor de la cuarta parte del espacio interior está ocupado por la fuente de alimentación eléctrica. Forrada de metal, su peso denota que no es una unidad de alimentación eléctrica avanzada de tipo conmutación, sino un sólido transformador de núcleo de hierro con condensadores. Puede parecer anticuado, pero tiene la ventaja de que es casi imposible sobrecargarlo o dañarlo.

Quizá sea esta fiabilidad lo que haya hecho tan popular el 380Z en el Ministerio de Defensa británico, que está empleando gran número de estas máquinas para el control de existencias y funciones similares. En aquellas escuelas y facultades que imparten matemáticas, física y ciencias a nivel más elevado, la máquina se ve especialmente favorecida en virtud de sus gráficos de alta resolución, que resultan de gran utilidad para demostraciones ilustradas de diversos puntos de los temarios de estudios.

El paquete de gráficos de alta resolución (HRG: *High Resolution Graphics*) es un conjunto de rutinas en código de lenguaje máquina que se llaman desde el programa del usuario y que pueden modificar la visualización generada por la ficha HRG. Ésta es imprescindible para producir cualquier gráfico; y, a pesar de que se introdujo hace algunos años, sigue siendo uno de los mejores sistemas existentes. Mediante la alteración del contenido de ciertas posiciones de memoria, la ficha puede generar visualizaciones en varias resoluciones con los colores normales (rojo, amarillo, verde, azul, magenta y cian). Según la resolución que se elija, que puede oscilar entre 160 × 96 y 320 × 192, a estos seis colores más el blanco se les pueden dar distintos niveles de brillo, aumentando de esta manera la gama de colores a 1 786 (siete veces 255, más el negro). Por otra parte, se pueden utilizar algunos de los bits normalmente destinados a especificar la densidad para producir múltiples páginas de gráficos, si bien la cantidad de intensidades diferentes será proporcionalmente inferior.



El monitor
Para aprovechar al máximo las facilidades del 380Z es esencial un monitor en color con una interface RGB. La máquina se puede comprar con 40 u 80 columnas en pantalla como estándar, lo que determina el tipo de monitor necesario

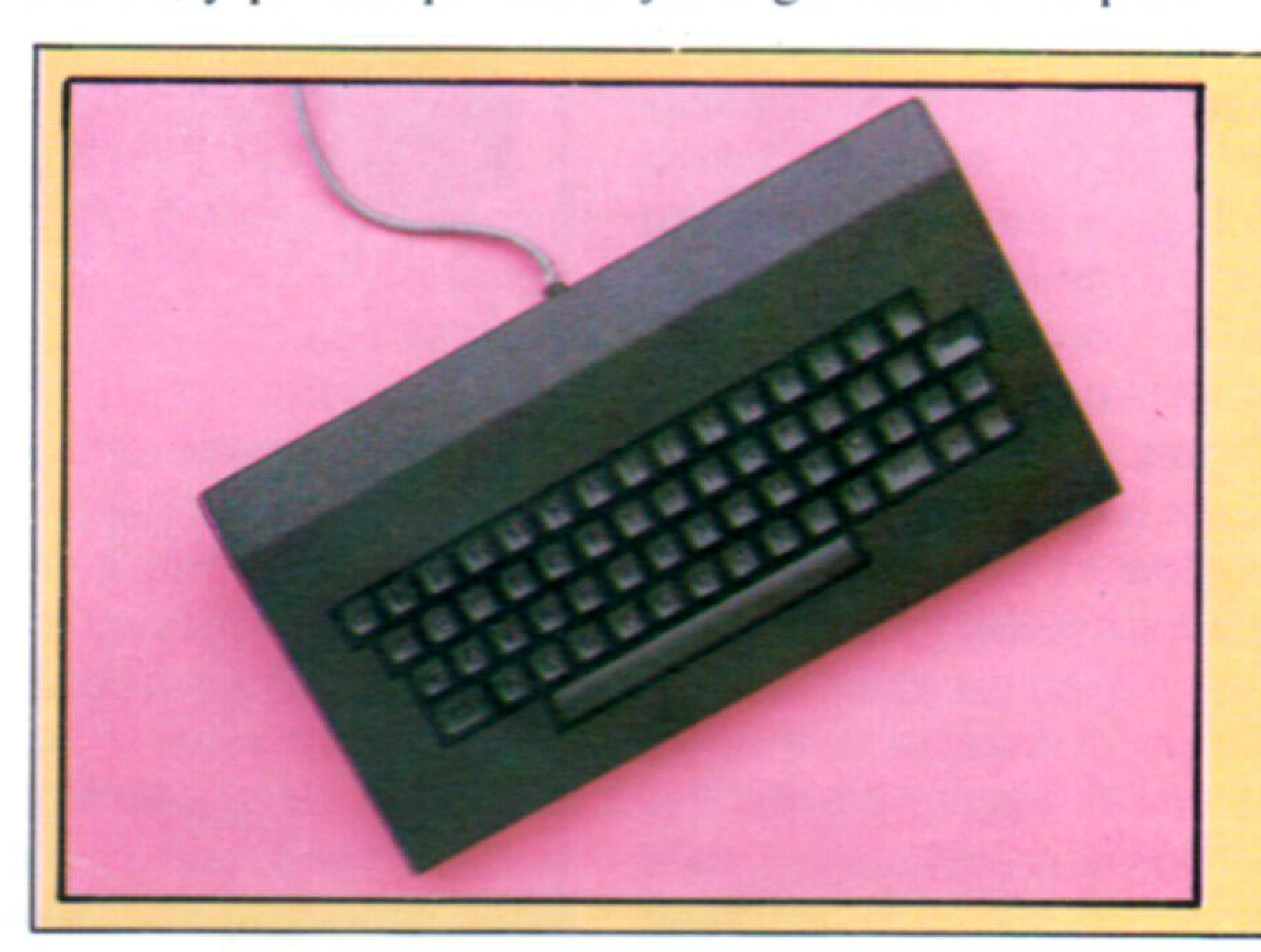
Chris Stevens

Tablero para control de la unidad de disco
Además de llevar un chip especializado para control de disco, este tablero posee un chip reloj-sincronizador Z80 (CTC) y un chip de input/output en serie 8521, que juntos proporcionan óptimas facilidades para comunicaciones

Tablero terminador de bus
Está situado en el extremo del bus más alejado del tablero de la CPU, y protege contra las interferencias a las diversas líneas eléctricas

Un juego completo de estas llamadas, más otras para manipular las cuestiones relacionadas con ellas, como *dump* de impresora (copiar la pantalla en el papel), se proporcionan como versiones ampliadas del BASIC RML. Esta versión se parece mucho al BASIC Microsoft y la mayoría de las palabras clave se emplean de forma idéntica. En este sentido, la única excepción importante la constituye el empleo de etiquetas con texto para llamar a las subrutinas, en vez de una dirección en decimal.

Sin embargo, el intérprete más el paquete HRG ocupan los dos una considerable cantidad de memoria, y puede que no dejen lugar suficiente para



El teclado
El teclado que se proporciona con el RML 380Z está montado en una caja metálica pequeña pero pesada. Las teclas se disponen de acuerdo a un patrón bastante estándar y son de gran calidad, con un tacto sólido pero agradablemente ligero. Es obvio que están diseñadas para resistir un uso intensivo, factor este que hace que el teclado sea ideal para la escuela



Tablero de la VDU
El generador de visualizaciones no forma parte del tablero principal, como en las máquinas más pequeñas, sino que es un dispositivo separado al que se le envían las señales de control de la CPU. Existen unidades de representación en video con 40 u 80 columnas

Tablero de la RAM
El tablero principal de la CPU no puede retener la RAM suficiente para proporcionar un sistema utilizable, de modo que se puede agregar más RAM, hasta el límite de direccionamiento del Z80, que es de 64 Kbytes

Conexiones del bus Z80

Tablero de la CPU
Lleva la CPU Z80, parte de la RAM, la ROM y la mayoría de los componentes esenciales que se requieren para hacer funcionar la máquina, tales como una puerta para el teclado

Tablero HRG
Al 380Z se le pueden agregar gráficos de alta resolución a todo color sencillamente enchufando este tablero

Fuente de alimentación eléctrica
Enorme, pesada y muy difícil de dañar

Mando de borrado

Cerradura
Permite cerrar la máquina para impedir que cualquier persona pueda interferir programas esenciales

Ventilador
Debido a que la fuente de alimentación eléctrica tiene la capacidad de manipular tableros adicionales sin ninguna modificación, se necesita un ventilador para ayudar a la refrigeración

Dos unidades de disco

Chris Stevens

Research Machines 380Z

DIMENSIONES
595 × 425 × 215 mm

CPU
Z80

VELOCIDAD DEL RELOJ
4 MHz

MEMORIA
Hasta 6 Kbytes de ROM
56 Kbytes de RAM

VISUALIZACION EN VIDEO
24 líneas de 40 u 80 caracteres, 7 colores con hasta 255 tonos. Resolución para gráiicos de 320 × 192 y 160 × 96

INTERFACES
RS232 en serie, cassette, impresora en paralelo

LENGUAJE SUMINISTRADO
BASIC Research Machines ampliado

OTROS LENGUAJES DISPONIBLES
ALGOL, FORTRAN y CP/M estándar

VIENE CON
Manuales de instalación, CP/M, sistema de disco, sistema de cassette y programas de utilidades BASIC en disco

TECLADO
60 teclas de calidad de procesador de textos

DOCUMENTACION
Excelente, aunque algo árida. La información es exhaustiva y resulta de fácil acceso para el usuario

programas de usuario sofisticados. Por esta causa se proporcionan tres versiones de BASIC, incluyendo todo el paquete HRG, parte del mismo o bien nada en absoluto, según las necesidades de memoria del usuario.

El empleo del paquete HRG no se limita al BASIC, pues funciona como un sencillo archivo en código máquina, al que se puede acceder a través de cualquier lenguaje. Dado que el 380Z por lo general opera bajo CP/M (véase p. 410), ello significa que la gama de lenguajes disponibles es amplia, pero el hecho de que la máquina posea una versión de ALGOL configurada para funcionar con ella, la convierte en algo insólito en el mundo del microordenador.

Este lenguaje, que muchos científicos europeos prefieren al FORTRAN (el lenguaje favorito para fines científicos en Estados Unidos), se asemeja al PASCAL y es especialmente eficaz en cálculos matemáticos complejos, como los que exige el diseño estructural. Éste es otro factor que hace que la máquina les resulte atractiva a los educadores.

Sonido ideal

El BASIC del Commodore 64 no responde al notable nivel de sus facilidades para sonido

Entre los ordenadores personales más populares, es el Commodore 64 el mejor equipado para la producción de sonido, y esto gracias a un chip especial denominado *Sound interface device* (dispositivo interface de sonido) o SID, que son sus siglas de brega.

El SID tiene unas posibilidades similares a las de un sintetizador monofónico comercial. Hay tres osciladores con una amplitud de ocho octavas (de 0 a 3900 Hz en 65 536 intervalos); un control principal de volumen, de 0 a 15; cuatro formas de onda, o timbres, para cada oscilador (triangular, aserrada, pulso de anchura variable y ruido); sincronización de osciladores y generadores de envolventes susceptibles de un control ADSR para cada oscilador. Otras configuraciones son: modulación circular; filtro programable con pasada baja, pasada de banda, pasada alta, de salida escalonada (que bloquea una banda estrecha de frecuencias) y resonancia variable; filtración de envolvente; dos interfaces para potenciómetro analógico-digital que se pueden emplear para controlar las facilidades del SID, y una salida de audio externa, que permite conectar entre sí chips SID adicionales. Con las salidas de SID estándar se puede dar entrada, filtrar y mezclar a otras señales de audio.

Sería imposible detallar el funcionamiento de cada una de estas configuraciones (existen libros muy buenos sobre la materia), pero lo que sí podemos hacer es explicar el significado de todos estos términos. En primer lugar, la sincronización de osciladores hace que dos señales (en este caso, dos voces especificadas) se acoplen entre sí armónicamente, produciendo a partir de las dos señales separadas un solo tono más complejo.

La modulación es la modificación de una señal por otra, que afecta o bien la frecuencia o la amplitud (volumen) del sonido. La modulación circular es la modulación de amplitud de una voz con otra, lo cual produce un tono que es claro pero que posee un efecto disonante y chillón; se puede utilizar para producir sonidos del tipo campana similares a los producidos por un bidón vacío. Se dice que estos sonidos poseen sobretonos inarmónicos.

Los filtros permiten eliminar de una señal escalas de frecuencia especificadas. Los diferentes tipos de filtración posibles en el Commodore 64 tienen los efectos que sugieren sus nombres: los filtros de pasada baja descartan las frecuencias que sean mayores que una frecuencia especificada; los filtros de pasada de banda eliminan las frecuencias por arriba y por debajo de una «banda» de frecuencias especificada; los filtros de salida escalonada son el opuesto de los filtros de pasada de banda: bloquean una

Luz espacial

Los gráficos Player-Missile son uno de los puntos fuertes de las máquinas Atari

Los gráficos Player-Missile

Los gráficos *Player-Missile* (jugador-misil) o "PM" constituyen un aspecto muy importante de las capacidades para gráficos de Atari. Su naturaleza es similar a la de los gráficos sprite que existen para el Commodore 64 (véase p. 408) y el Sord M5, permitiendo al programador el diseño y control de hasta ocho figuras diferentes en alta resolución. Estas figuras móviles operan con independencia de la visualización del fondo y se las puede programar para que se desplacen por delante o por detrás de otras figuras que pueden estar dibujadas en la pantalla. Ello le permite al programador incorporar una tercera dimensión en los efectos de pantalla. Los gráficos PM se pueden mover suave y velozmente por la pantalla, y por ello son ideales para los juegos recreativos de acción rápida. También son útiles para crear visualizaciones estáticas más coloridas que las que se consiguen empleando las modalidades normales para gráficos, porque los objetos PM se pueden colorear por separado e independientemente de la visualización del fondo.

Al igual que con todos los gráficos sprite, el secreto de las facilidades para los gráficos PM radica en un hardware *ad hoc*. Hay registros especiales diseñados para controlar el movimiento, el color y la visualización en pantalla de los objetos PM. Todo cuanto el programador debe hacer es colocar ciertos valores en estos registros para manipular los objetos. En BASIC, esto se realiza mediante la orden POKE. Una vez que se ha colocado (POKE) un número en el registro adecuado, el propio hardware de Atari se hace cargo del resto del trabajo. Esto se efectúa a la velocidad del código de lenguaje máquina y, por consiguiente, mucho más rápido que si el proceso se controlara mediante BASIC.

Analicemos ahora el proceso para crear objetos PM y los registros para controlarlos. Los jugadores se diseñan a partir de una franja vertical, de ocho pixels de ancho y 128 o 256 de alto. Cada una de las filas de la franja se representa en la memoria del ordenador como un único byte. Mediante la colocación (POKE) de los códigos binarios adecuados, se puede definir la forma de un jugador utilizando un método similar al que se emplea para crear caracte-

```
10 SID = 54272
20 POKESID + 23,0
30 POKESID + 24,15
40 POKESID + 5,40
50 POKESID + 6,201
60 FOR N = 1TO5
70 READ FH,FL,D
80 POKESID + 1,FH:
   POKESID,FL:
   REM*TOCAR
   NOTA*
90 POKESID + 4,33
100 FORI = 1TO300*D:
    NEXT I
110 POKESID + 4,32
120 FORI = 1TO100:
    NEXT I
130 NEXT N
140 FORI = 1TO2000:
    NEXT I
150 POKESID + 24,0
160 REM**FH FL D**
170 DATA 57,172,1
180 DATA 64,188,1
190 DATA 51,97,1
200 DATA 25,177,1
210 DATA 38,126,2
220 END
```

banda específica; los filtros de pasada alta cortan las frecuencias que sean inferiores a una especificada; y la resonancia variable se puede aplicar a todos los filtros anteriores para enfatizar las frecuencias alrededor de los puntos de corte. La filtración de envolvente constituye un caso especial: su efecto difiere de las otras en que los valores digitalizados de ADSR establecidos para la envolvente 3 se pueden leer desde el chip SID y aplicar a una señal, de modo tal que la estructura armónica se vaya modificando en el transcurso de una nota. Funciona como un filtro variable.

Estas múltiples configuraciones permiten construir, con sonidos sumamente complejos, interesantes efectos y convincentes imitaciones de instrumentos convencionales. La faceta desalentadora del SID es que el BASIC CBM V2, la versión con que viene el 64, no dispone en absoluto de ninguna orden directa de sonido. Éste se obtiene utilizando PEEK y POKE para los 29 registros de control del SID. Por consiguiente, se requiere muchísima codificación para generar hasta los efectos más sencillos y, en algunos casos, el BASIC no es lo suficientemente veloz como para hacer justicia a la gama completa de posibilidades del SID.

No podemos describir a fondo los registros de control del SID, pero, con lo dicho, puede usted tocar notas agradables, con programas como el de la izquierda.

Aunque la longitud del programa es de 22 líneas, sólo toca cinco notas de una melodía sencilla en un oscilador. La línea 20 desconecta el filtro de los osciladores; la línea 30 establece en el máximo el volumen principal; y las líneas 40 y 50 especifican una envolvente similar al piano. La línea 80 establece la frecuencia de las notas; la 90 y la 100 empiezan y terminan el ciclo ADSR y seleccionan una onda aserrada para la voz 1; y la sincronización para los bucles FOR...NEXT se obtiene mediante las líneas 100, 120 y 140.

Programar sonido en BASIC en un Commodore 64 requiere gran esfuerzo tanto para aprender el código como para escribirlo. Incluso puede ser un ejercicio desmoralizador, porque la única manera de descubrir si un conjunto de sentencias, ya de por sí complicadas, funciona a un compás aceptable, es mediante el ensayo y el error. Si usted desea métodos más sencillos para generación de sonido, vale la pena investigar los muchos programas para edición de sonido que existen a la venta en el mercado. Éstos suelen estar escritos en lenguaje máquina, y aprovechan al máximo las maravillosas configuraciones del Commodore 64.

res definidos por el usuario (véase p. 246). De esta manera se pueden definir hasta cuatro jugadores, ocupando cada uno de ellos sus 256 o 128 bytes de memoria.

Cada uno de los cuatro jugadores posee una figura de misil relacionada con él cuya anchura es de dos bits. Para crear jugadores y misiles es necesario colocar (POKE) los patrones de bits que definen su forma en una zona determinada de la memoria. La zona de RAM utilizada la puede elegir el programador, pero se le debe informar de ello al ordenador estableciendo un indicador al principio de la zona.

Si el programador opta por emplear una resolución vertical de un solo pixel, entonces se necesita el doble de la memoria requerida para una resolución vertical de dos pixels. El siguiente programa diseña al jugador 0 en resolución vertical de dos pixels como una nave espacial:

```
10 REM ***DEFINIR UN JUGADOR***
20 P = PEEK(106) -8: REM ESTABLECE EL P EN
   2K POR DEBAJO DE LA PARTE SUPERIOR DE
   RAM
30 POKE 54279,P: REM ESTABLECE PUNTERO EN
   ZONA PM
40 BASE = 256*P:REM ESTABLECE LA DIRECCION
   DE BASE ZONA PM
50 FOR I = BASE + 512 TO BASE + 640
60 POKE I,0: REM LIMPIA ZONA JUGADOR 0
70 NEXT I
80 FOR I = BASE + 512 + 50 TO
   BASE + 530 + 50
90 READ A: POKE I,A: REM DEFINIR FIGURA
100 NEXT I
110 DATA 16,16,16,56,40,56,40,56,40
120 DATA 56,56,186,186,146,186,254,186,146
```

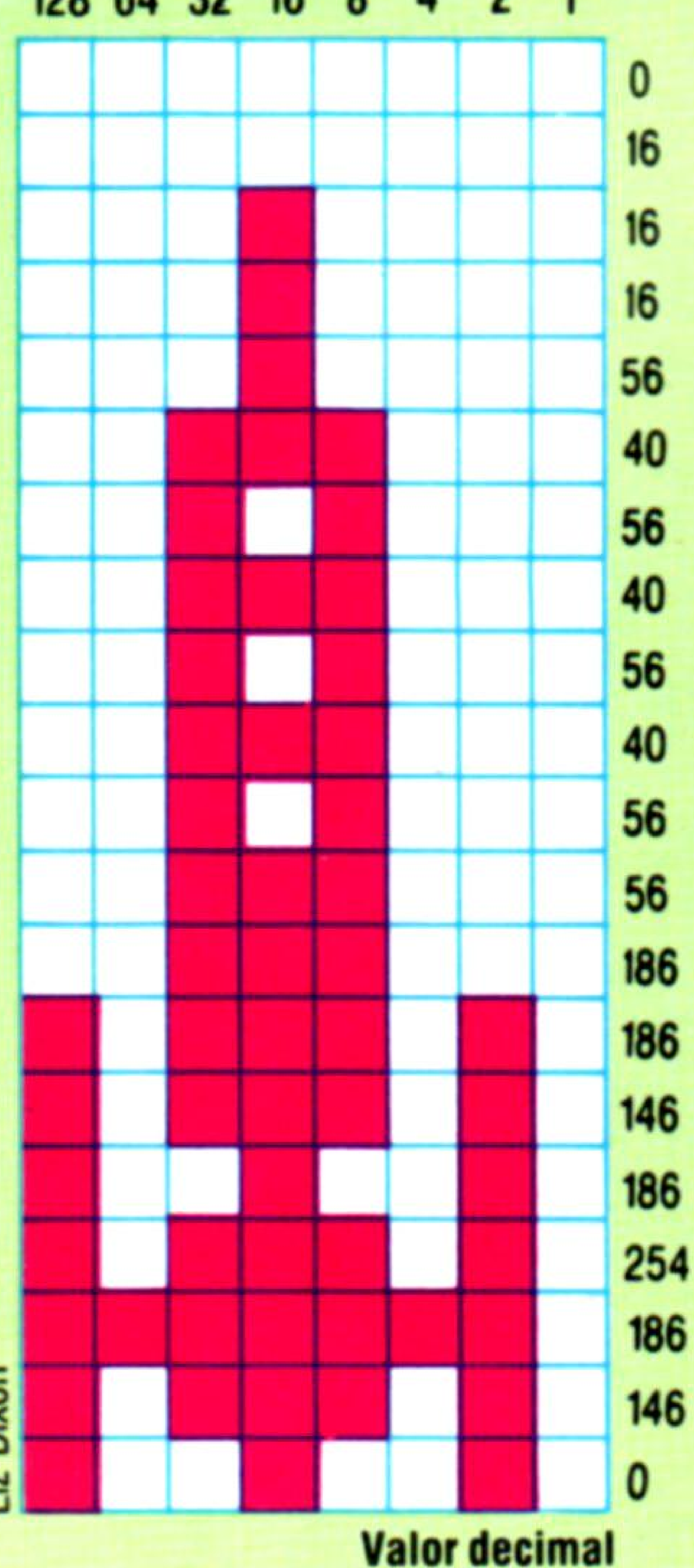
Cada figura de jugador tiene varios registros relacionados con ella. Estos registros controlan el color, la posición horizontal y el tamaño. El último permite que el programador multiplique el ancho de un jugador por el factor dos o el cuatro. Otros registros controlan la prioridad del jugador con respecto al fondo. Los misiles toman el color del jugador que los guía, pero su tamaño se puede modificar independientemente. Para las aplicaciones de juegos se dispone de una serie de registros que detectan las colisiones en pantalla entre los jugadores, los misiles y el fondo. No obstante, no hay registro de posición vertical ni para misiles ni para jugadores. El movimiento vertical de un jugador se obtiene elevando, a través de la zona de la memoria reservada a ese jugador, el contenido de cada una de las posiciones que guardan los patrones de bits para la figura. En lenguaje ensamblador esta tarea es bastante directa, pero en BASIC es lenta.

Los gráficos Player-Missile amplían considerablemente el potencial para gráficos de Atari, aunque no son tan versátiles ni fáciles de utilizar como los sprites del Commodore 64. Ofrecemos una continuación del programa que empezamos anteriormente, para colorear la nave espacial y desplazarla de izquierda a derecha a través de la pantalla.

```
130 POKE 559,46: REM VISUALIZA LINEA 2 PM
140 POKE 53277,3: REM VISUALIZA PM
150 POKE 704,88: REM COLOREA ROSA
    JUGADOR 0
160 GRAPHICS 0
170 SETCOLOUR 2,8,2: REM ESTABLECE FONDO
    AZUL OSCURO
180 FOR I = 0 TO 320
190 POKE 53248,I: REM ESTABLECE POSICION
    HORIZONTAL
200 NEXT I
210 END
```

Cohete PM
Antes de definir una pieza de juego hay que dibujarla, calculando después los valores decimales para cada una de las filas de pixels

Franja para el jugador



Laboratorio de idiomas

Para resumir lo hasta aquí explicado, analicemos con ojo crítico el lenguaje BASIC y algunas de las alternativas al mismo

Vamos a analizar brevemente los puntos fuertes y los puntos débiles del BASIC respecto a los otros lenguajes de programación.

El BASIC deriva del FORTRAN, uno de los primeros lenguajes de programación. A diferencia de la mayoría de los otros lenguajes, el BASIC se interpreta. Esto significa que cuando se ejecuta un programa otro programa especial, situado en algún lugar de la memoria del ordenador, capta el código sentencia a sentencia, y lo convierte en código de lenguaje máquina. He aquí lo que sucedería con un breve programa en BASIC como éste:

```
10 CLS
20 PRINT "DIGITE UN NUMERO"
30 INPUT X
40 PRINT "DIGITE UN SEGUNDO NUMERO"
50 INPUT Y
60 "PRINT" "El PRODUCTO DE LOS DOS NUMEROS
   ES: ";
70 PRINT X*Y
80 PRINT
90 PRINT "¿DESEA OTRA PRUEBA?"
100 PRINT "PULSE 'S' PARA PROBAR OTRA VEZ"
110 PRINT "O BIEN 'N' PARA TERMINAR"
120 FOR X = 1 TO 1
130 LET A$ = INKEY$
140 IF A$ <> "S" AND A$ <> "N" THEN X = 0
150 NEXT X
160 IF A$ = "S" THEN GOTO 10
170 END
```

Cuando el intérprete de BASIC se encuentra con la línea 10, elabora el código de lenguaje máquina necesario para limpiar la pantalla. En la línea 20 prepara en código de lenguaje máquina las instrucciones necesarias para enviar a la pantalla el mensaje DIGITE UN NUMERO. En la línea 30, determina el espacio de memoria necesario para almacenar un número real, espera la entrada desde el teclado y después convierte el número digitado a binario para almacenarlo en el espacio destinado a la variable X. Todo esto se repetiría para las líneas desde la 40 a la 60. Si el usuario deseara repetir el programa, digitando S, el intérprete retrocedería hasta la línea 10, y repetiría otra vez todos los cálculos.

Por el contrario, la mayoría de los demás lenguajes se "compilan". Ello significa que después de que se ha escrito un programa, para que se lo pueda ejecutar, antes lo debe procesar un *compilador*. El compilador es un programa independiente que examina detenidamente el "código fuente" (el programa original) y produce una segunda versión del mismo en código de lenguaje máquina. Cuando se ejecuta el programa compilado, es lógico que funcione mucho más rápido que un programa interpretado, porque ya se han efectuado todas las traducciones a lenguaje máquina, que requieren bastante tiempo.

Si los programas compilados son más rápidos que los interpretados, puede que se pregunte por qué razón no todos los lenguajes de programación utilizan compiladores. Varias son las ventajas que ofrecen los programas interpretados, como el BASIC. La mayoría de éstas derivan del hecho de ser un lenguaje interactivo, es decir, que se puede verificar y depurar "frente al teclado", mientras se está desarrollando el programa. El BASIC, por ejemplo, admite la inserción de la orden STOP en cualquier punto del programa. En el momento en que el intérprete la encuentra, deja inmediatamente de interpretar el programa y permite que se impartan "órdenes" desde el teclado.

Las órdenes son instrucciones que el intérprete puede ejecutar directamente cuando no se está ejecutando el programa. El BASIC posee una gran cantidad de órdenes, y éstas pueden ser de enorme valor para la depuración. Después de que un programa en BASIC se ha ejecutado (es decir, después de que el intérprete se ha encontrado con la sentencia END), o cuando el intérprete se encuentra con una sentencia STOP, se pueden imprimir (PRINT) los valores de todas las variables. Intente ejecutar el programa de la agenda de direcciones, por ejemplo. Ejecute el programa y digite 9 para salir del mismo. Si funciona de principio a fin sin que aparezca ningún mensaje de error, debería acabar con una muletilla típica (por lo general, OK, > o *). Luego, digite PRINT RMOD <CR>. El intérprete debe imprimir un 0 en la pantalla (¡siempre y cuando no se hubiera agregado ningún registro!). Luego pruebe PRINT TAMA <CR>. La consecuencia será que el intérprete imprimirá en la pantalla un número superior en una unidad al número de registro que usted tenga en el archivo de datos.

Ventajas del BASIC

Se suele decir que el BASIC es el lenguaje ideal para el programador sin experiencia, pues permite eliminar los *bugs* desde el teclado. Posee otra importante ventaja: es comparativamente fácil de aprender. Por ejemplo, en esta veintena de capítulos dedicados a la programación BASIC hemos tocado todos los puntos fundamentales y muchos de los aspectos avanzados del BASIC en apenas 86 páginas. Los errores sintácticos tales como 40 PRINT A(12) provocan por lo general mensajes de error de fácil comprensión al ejecutar el programa, como SYNTAX ERROR IN 40. Con sólo una mirada al número de línea aludido adivinamos dónde está el error, y con la misma facilidad se elimina mediante EDIT 40 <CR> (seguido de unas pocas órdenes de edición) o volviendo a digitar la línea del modo correcto. Siempre que el intérprete de BASIC se encuentra con un error de sintaxis o de lógica, interrumpe la ejecución del programa e informa del error. Eliminar

los *bugs* es tan sencillo como probar con una nueva línea en sustitución de la equivocada y digitar RUN <CR> otra vez.

Desventajas del BASIC

Algunas desventajas son muy sutiles, pero las hay flagrantes. Dado que se interpreta (hay sólo unas pocas versiones compiladas de BASIC), funciona con mucha lentitud. Si la velocidad no es tan decisiva (p. ej., en un programa para calcular el saldo de su cuenta bancaria), la lentitud del BASIC interpretado es soportable. Pero si la velocidad resulta esencial (como es el caso de un programa para animación en pantalla que utilice gráficos, o cuando se trata, por ejemplo, de un "reloj" empleado para medir las reacciones en un experimento de laboratorio), hay muchas probabilidades de que el BASIC interpretado resulte demasiado lento.

Si usted necesita velocidad en sus programas tiene dos caminos a seguir: programar en código de lenguaje máquina o bien en lenguaje ensamblador (véase p. 448), que es difícil y laborioso, o programar en un lenguaje compilado como el PASCAL o el FORTH. Los lenguajes compilados no son difíciles de aprender, pero el código fuente (el programa original) casi con toda seguridad contendrá errores, que el compilador descubrirá cuando intente compilar el programa. Éstos son difíciles de rectificar, en comparación con los errores del BASIC. Una vez que se ha corregido el código fuente, el programa se tendrá que volver a compilar todo de nuevo. La mayoría de los compiladores dan dos o tres "pasadas" por el código fuente, y es probable que cada una de estas sucesivas pasadas produzca mensajes de error, cada uno de los cuales, a su vez, se habrá de corregir antes de que el programa se pueda volver a compilar.

Realizar un programa correctamente compilado puede resultar un proceso que requiera más tiempo que obtener un programa en BASIC interpretado. Por añadidura, el BASIC puede "viciar" al programador novato, a quien, ahorrándole la puerta estrecha, le permitirá malas técnicas de programación que lenguajes altamente estructurados como el PASCAL rechazarían. El BASIC tolera programas muy poco cuidados, llenos de sentencias GOTO, por ejemplo, y estos malos hábitos arraigados pueden llegar a dificultar de forma notoria la transición a lenguajes más avanzados.

Y después del BASIC, ¿qué?

El BASIC es un lenguaje flexible no difícil de aprender. Posee excelentes facilidades para manipulación de variables, pero es lento y no consigue sacar el máximo partido del potencial de un ordenador personal. Por otra parte, lenguajes más modernos, como el PASCAL y el FORTH, ofrecen facilidades de programación que en BASIC son difíciles o bien imposibles.

El PASCAL también se ideó como lenguaje de enseñanza y se diseñó específicamente para favorecer el desarrollo de programas bien construidos, "estructurados". El PASCAL es un lenguaje compilado, lo que significa que los usuarios se pueden encontrar con numerosos errores captados por el compilador (después de que se ha escrito el código fuente y antes de que se pueda ejecutar el código objeto compilado), y esto puede ser muy frustrante. Los programadores poco expertos en PASCAL hallan además en los requerimientos del lenguaje, tales como la necesidad de definir todas las variables al comienzo del programa (y no sólo eso, sino también la precisión de tener que definir de qué tipo son, reales, enteras, etc.), un impedimento para la programación libre y ágil.

Por otra parte, el PASCAL exige que el programador clarifique la lógica del programa antes de escribirlo. Es probable que los programas en este lenguaje arrojen numerosos errores sintácticos en el código fuente, pero también cabe la posibilidad —que incluso puede decirse que es más probable— de que estén bien diseñados, en cuyo caso difícilmente contendrán errores fundamentales de lógica.

Últimamente el FORTH se ha convertido en una alternativa muy popular al BASIC como lenguaje de programación para micros personales. Aunque aprender FORTH no es tan difícil como aprender lenguaje ensamblador o código de lenguaje máquina, es menos "intuitivo" que el BASIC o el PASCAL. Aun así, el FORTH posee muchos méritos exclusivos que hacen de él un buen candidato a ser el segundo lenguaje aconsejable para un programador.

A pesar de que el FORTH es un lenguaje de alto nivel, funciona casi tan rápido como el código de lenguaje máquina, debido a la forma exclusiva en que trabaja. Mientras que un lenguaje como el BASIC posee un número fijo de sentencias y órdenes, los usuarios de FORTH pueden definir su propio vocabulario.

La palabra clave PRINT en BASIC significa que cualquier carácter que venga a continuación entre comillas dobles se imprimirá en la pantalla. El programador no puede alterar este hecho. En FORTH, PRINT se podría definir para que haga, por ejemplo, un listado en la pantalla, impreso en una columna vertical, de los equivalentes hexadecimales de los códigos ASCII correspondientes a los caracteres que componen una variable.

El FORTH le otorga al programador el poder de definir cualquier palabra para significar lo que desee y para producir los resultados deseados cada vez que sea empleada a partir de entonces. Es no sólo sumamente flexible en este sentido, sino que ofrece al usuario la ventaja adicional de que también produce programas que se pueden compilar en código objeto (véase p. 184), que son casi tan compactos y de rápido funcionamiento como los programas en lenguaje máquina.

Aunque existen muchos lenguajes de programación, la mayoría de los aficionados se inclinarán, después del BASIC, a escoger entre el lenguaje ensamblador, el PASCAL y el FORTH. Muy brevemente sintetizaremos a continuación las ventajas y las desventajas de cada uno de ellos:

BASIC
Fácil de aprender
Fácil de recordar
Fácil de depurar
De lenta ejecución
Ocupa mucha memoria
No favorece la programación estructurada

Lenguaje ensamblador
No tan fácil de aprender
No tan fácil de recordar
Difícil de depurar

De muy rápida ejecución
Otorga al usuario un control completo sobre el microprocesador

PASCAL
Bastante fácil de aprender
Bastante fácil de recordar
Más difícil de depurar que el BASIC
Favorece mejores técnicas de programación
De ejecución más rápida que el BASIC y más lenta que el ensamblador
Necesita compilarse, lo que lleva su tiempo; después de compilado, funciona casi tan rápidamente como el ensamblador
Otorga bastante control sobre el microprocesador, pero menos que el ensamblador; el manejo de variables no es tan sencillo como en BASIC

FORTH
No resulta muy fácil de aprender; más fácil de asimilar por los principiantes absolutos que por programadores en BASIC
Bastante fácil de recordar
Muy fácil de depurar en modalidad de intérprete
Se puede compilar; se ejecuta casi tan rápidamente como el lenguaje ensamblador
Otorga control completo sobre el microprocesador
Economiza memoria
Más fácil de aprender que el lenguaje ensamblador, aunque menos "intuitivo" que el BASIC

Complementos al BASIC




```
1 REM *CREAR ARCHIVO DE DATOS*
2 DIM N$(30)
3 LET N$ = "@VACIO"
4 DIM F$(15)
5 LET F$ = "FICTICIO"
6 LET Z = 2
7 EXTBACK 1
8 EXTSTORE 1,Z,N$,N$,N$
9 EXTSTORE 1,F$,F$,F$,F$
10 INPUT "INSERTAR CINTA DE DATOS,
   PULSAR RECORD, Y DIGITAR 'Y'";A$
11 SSAVE 1, "DAT.AGCO"
12 PRINT "PARAR LA CINTA Y
   REBOBINAR"
13 END
```

Nota: Éste es el programa de inicialización para el Lynx de 96 K; carecemos de información acerca de la manipulación de archivos en cinta para los otros modelos.

Variables del programa principal
Copiar el listado para el Spectrum con estas sustituciones para las variables numéricas:

Cambiar	TAMA	por Z
	RMOD	por R
	CLAR	por D
	CURS	por C
	OPCN	por O
	INF	por i
	MED	por m
	SUP	por s

y efectuar las siguientes modificaciones, sustituciones y supresiones de líneas:

```
1100 REM S/R *CREMAT*
1110 DIM N$(30)(50)
1120 DIM M$(30)(50)
1130 DIM C$(30)(50)
1140 DIM D$(15)(50)
1150 DIM P$(15)(50)
1160 DIM T$(15)(50)
1170 DIM X$(15)(50)
1180 DIM Z$(30)
1210 LET Z = 0
1220 LET R = 0
1230 LET D = 1
1240 LET C = 0
1250 LET Z$ = "@VACIO"
1260 LET Q$ = ""
1300 RETURN
.
1400 REM S/R *LEARCH*
1405 PRINT "INSERTAR CINTA DE DATOS Y
     PULSAR PLAY"
1410 GOSUB 3100
1420 SLOAD 1, "DAT·AGCO"
1430 PRINT "PARAR LA CINTA"
1440 GOSUB 3100
1450 EXTBACK 1
1460 EXTFETCH 1,Z
1470 FOR K = 1 TO Z-1
1480 EXTFETCH 1,
     N$(K),M$(K),C$(K),D$(K),P$(K),T$(K),
     X$(K)
1490 NEXT K
1500 LET Q$ = N$(1)
1510 RETURN
.
3120 IF KEYN <> 32 THEN LET L = 0
.
3780 LET A$ = KEY$
.
3810 LET O = VAL(A$)
3820 IF (O < 1) OR (O > 9) THEN LET
     L = 0
.
4500 REM S/R *MODNOM*
4510 REM CONVERSION A MAYUSCULAS
4520 LET R$ = UPC$(N$(Z))
     (suprimir desde línea 4530 a 4590)
.
4600 LET S$ = ""
4601 LET A$ = ""
4602 LET T = LEN(R$)
4603 LET S = 0
.
4610 REM LOCALIZAR ULTIMO ESPACIO
.
4630 IF MID$(R$,L,1) = " " THEN LET
     S = L
.
4670 IF MID$(R$,L,1) > "@" THEN LET
     S$ = S$ + MID$(R$,L,1)
.
4710 IF MID$(R$,L,1) > "@" THEN LET
     A$ = A$ + MID$(R$,L,1)
```

Desde la 5410 a la 5460, las líneas se deben reducir a sentencias simples, por ejemplo:

```
5410 LET U$ = N$(L):LET
     N$(L) = N$(T):LET
     N$(T) = U$ se convierte en
5410 LET U$ = N$(L)
5411 LET N$(L) = N$(T)
5412 LET N$(T) = U$
```

y así sucesivamente, sin otros cambios.

```
5600 REM S/R *GRDREG*
5605 PRINT "INSERTAR CINTA DE DATOS
     Y PULSAR RECORD"
5610 GOSUB 3100
5620 EXTBACK 1
5630 EXTSTORE 1,Z
5640 FOR K = 1 TO Z-1
5650 EXTSTORE 1,
     N$(K),M$(K),C$(K),D$(K),P$(K),T$(K),
     X$(K)
5660 SSAVE 1, "DAT-AGCO"
5670 PRINT "PARAR LA CINTA"
5680 GOSUB 3100
5690 RETURN

5855 LET X = 0
5860 LET m = INT((i + s)/2)
5870 IF M$(m) = U$ THEN LET X = 1
```

```
5880 IF U$ > M$(m) THEN LET i = m + 1

6080 LET A$ = KEY$
6110 IF A$ = " " THEN RETURN
6120 GOSUB 6200
6130 RETURN

6730 FOR I = 1 TO 1
6735 LET I = 0
6740 LET A$ = KEY$
6750 IF (A$ = E$) OR (A$ = " ") THEN
     LET I = 1
6760 NEXT I
```

Este fragmento se debe reproducir en las líneas 6880-6910, 6990-7030, 7110-7140, 7220-7250, 7640-7670

DRAGON 32

BBC MICRO

COMMODORE 64

VIC-20

Programa de inicialización
Éste es el programa de inicialización para el Dragon 32.

```
1 REM *CREAR ARCHIVO DE DATOS*
2 LET Z = 2
3 LET N$ = "@VACIO"
4 OPEN "0", #-1, "DAT-AGCO"
5 INPUT "INSERTAR CINTA DE DATOS,
  PULSAR RECORD Y DIGITAR 'Y'";A$
6 PRINT# - 1,Z,N$,N$,N$,N$,N$,N$,N$
7 CLOSE# - 1
8 PRINT "PARAR LA CINTA Y REBOBINAR"
9 STOP
```

En el BBC Micro, sustituir las líneas 4, 6 y 7 por:

```
4 F1 = OPENOUT("DAT-AGCO")
6 PRINT #F1,Z,N$,N$,N$,N$,N$,N$,N$,
7 CLOSE #F1
```

En el Commodore 64 y en el Vic-20, reemplazar las líneas 4, 6 y 7 por:

```
4 OPEN 1,1,2,"DAT-AGCO"
6 PRINT #1,Z,N$,N$,N$,N$,N$,N$,N$,
7 CLOSE 1
```

Variables del programa principal
En el Dragon, los Commodore y el BBC Micro, copiar el listado del Spectrum (publicado completo en p. 458) con estas sustituciones para las variables numéricas.

Cambiar:		
	TAMA	por Z
	RMOD	por R
	CLAR	por D
	CURS	por C
	OPCN	por O
	INF	por IN
	MED	por MD
	SUP	por SP

y efectuar las siguientes modificaciones, sustituciones y supresiones de líneas:

```
1100 REM S/R *CREMAT*
1110 DIM N$(50)
1120 DIM M$(50)
1130 DIM C$(50)
1140 DIM D$(50)
1150 DIM P$(50)
1160 DIM T$(50)
1170 DIM X$(50)
```

Eliminar las líneas 1180-1190

```
1210 LET Z = 0
1220 LET R = 0
1230 LET D = 1
1240 LET C = 0
1250 LET Z$ = "@VACIO"
1260 LET Q$ = ""
1300 RETURN
```

Ésta es la versión de la subrutina 1400 para el Dragon 32:

```
1400 REM S/R *LEARCH*
1410 OPEN"I", #-1, "DAT-AGCO"
1420 PRINT "INSERTAR CINTA DE DATOS Y
     PULSAR PLAY"
1430 GOSUB 3100
1440 INPUT #-1,Z
1450 FOR K = 1 TO Z-1
1460 INPUT # - 1,1N$(K),M$(K),C$(K),
     C$(K),D$(K),P$(K),T$(K),X$(K)
1470 NEXT K
1480 Q$ = N$(1)
1490 CLOSE #-1
1500 PRINT "PARAR LA CINTA"
1510 GOSUB 3100
1520 RETURN
```

En el listado anterior, para el BBC Micro reemplazar la línea 1410 por:

```
1410 F1 = OPENIN("DAT-AGCO")
```

y sustituir #-1 por #F1 en las líneas 1440, 1460 y 1490

En el listado anterior, para el Commodore 64 y el Vic-20 reemplazar la línea 1410 por:

```
1410 OPEN 1,1,0,"DAT.AGCO"
```

Reemplazar #-1 por #1 en las líneas 1440 y 1460, y sustituir la 1490 por:

```
1490 CLOSE 1
```

En el BBC Micro, reemplazar INKEY$ por INKEY$(0) en todos los casos; y sustituir INPUT"...mensaje...";A$ por INPUT"...mensaje...",A$. En los Commodore, reemplazar LET A$ = INKEY$ por GET A$ en todos los casos; y sustituir IF INKEY$... por GET GT$:IF GT$...

En el BBC Micro, el Dragon y los Commodore, en la subrutina 4500 sustituir todas las referencias a R$(L) por MID$(R$,L,1). Reemplazar CODE... por ASC(...). Reemplazar PRIMER por ULTIMO en la línea 4610.
Suprimir: LET L = T del final de la línea 4630.

Ésta es la versión para el Dragon de la subrutina 5600; para las variaciones relativas al BBC y al Commodore, véanse las notas anteriores sobre el programa de inicialización.

```
5600 REM S/R *GRDREG*
5610 OPEN"0",#-1,"DAT-AGCO"
5620 PRINT "INSERTAR CINTA DE DATOS Y
     PULSAR RECORD"
5630 GOSUB 3100
5640 PRINT #-1,Z
5650 FOR K = 1 TO Z-1
5660 PRINT # - 1,Z,N$(K),M$(K),C$(K),
     D$(K),P$(K),T$(K),X$(K)
5670 NEXT K
5680 CLOSE #-1
5690 PRINT "PARAR LA CINTA"
5693 GOSUB 3100
5695 RETURN
```

En el BBC Micro, en la subrutina 6200 insertar:

```
6205 VDU 2
6275 VDU 3
```

y sustituir LPRINT por PRINT.

En los Commodore, insertar:

```
6205 OPEN 4,4:CMD 4
6275 PRINT #4:CLOSE 4
```

y sustituir LPRINT por PRINT.

En el Dragon, reemplazar LPRINT por PRINT #-2.

Cimientos sólidos

En la historia del microordenador, los avances en hardware y software van parejos, pues en ellos cuentan por igual tanto los productos como las personas

Ha habido momentos en la historia en que la magnitud del cambio tecnológico ha apabullado a la gente. Pero hasta la fecha, nada (ni siquiera la navegación aérea, desde los hermanos Wright hasta la exploración lunar) es comparable, en ritmo de progreso, a la revolución de la microelectrónica. El salto de los primitivos microprocesadores a los diseños actuales de 16 bits, de los primeros microsistemas a los actuales ordenadores de consola, se ha realizado en apenas una década. Y la velocidad de los avances es uniformemente acelerada.

Por el año 1971, varias de las nuevas firmas fabricantes de chips de California llegaron a la conclusión de que las principales funciones de un ordenador se podían alojar en una sola brizna de silicio. En aquel entonces no se intuía revolución alguna, ni se hablaba de ninguna "tecnología de la información". La idea se limitaba a producir un ordenador pequeño y barato que se pudiera utilizar para controlar las máquinas de las fábricas o los ascensores, y los primeros microprocesadores fueron muy adecuados para esas tareas.

Uno de estos fabricantes de chips, Intel, pasa por ser el inventor del primer microprocesador, denominado 4004. Los "cuatros" de la cifra aluden a su potencia: se trataba de un procesador de cuatro bits que manipulaba los datos en bloques de cuatro dígitos binarios. Estaba provisto de poca memoria, la suficiente como para albergar el programa de control de un ascensor, por ejemplo.

Hacia 1972 Intel había creado el chip 8008, un procesador de ocho bits, y los aficionados comenzaron a pensar en construirse ordenadores por sí mismos basándose en el nuevo chip. En las revistas norteamericanas para aficionados a la electrónica aparecían artículos describiendo cómo hacerlo, y aunque los ordenadores resultantes no poseían pantallas de monitor, ni teclados adecuados ni ningún otro medio auxiliar más evolucionado, aquéllas fueron las primeras máquinas personales. A partir de uno de estos proyectos para aficionados nació lo que podría considerarse el primer microordenador personal comercial que salió al mercado, el Altair-8800. No obstante, sólo se vendía como *kit* (es decir, como juego para montar).

Al año siguiente apareció el primer microprocesador "verdadero", el 8080, también producido por Intel. Operaba sobre bloques de datos de ocho bits y podía manipular hasta 64 Kbytes de memoria para programas más extensos. El reto fue recogido por las otras firmas de chips, que apretaron el paso. El chip 6800 de Motorola operaba de forma bastante similar al 8080. Las características de hardware eran parecidas, pero para hacerlo funcionar se requerían distintas instrucciones. Y aquí empezaron a suscitarse los problemas de la compatibilidad de software: los programas escritos para el 8080 no se podían ejecutar con el 6800, y viceversa.

Por la misma época, otras empresas estaban sacando al mercado procesadores similares, entre ellas National Semiconductor y Signetics and Advanced Micro Devices. Pero la impulsora principal de turno habría de ser MOS Technology, donde trabajaba uno de los protagonistas de nuestra historia, Chuck Peddle (véase p. 180). Peddle estaba en MOS Technology cuando la empresa creó un procesador muy parecido al 6800 de Motorola, denominado 6500. De hecho, era tan similar al 6800 que hubo que efectuar algunas modificaciones y al chip revisado al fin se le dio el nombre de 6502.

Los padres del invento
Aunque Chuck Peddle diseñó tanto el Commodore PET como el microprocesador 6502 en el cual se basaba, la contribución de Bill Gates, autor del BASIC Microsoft que está incorporado en la ROM del PET, fue igualmente importante



Cortesía de Commodore

Chuck Peddle

Tony Sleep

Bill Gates

Commodore ya era una empresa muy conocida en Canadá en maquinaria para oficina y calculadoras electrónicas. Peddle se incorporó a la empresa con la idea de producir un ordenador personal completo, con pantalla, teclado, cassettes para el almacenamiento de programas y todo aquello que debía poseer un verdadero ordenador; todo, por supuesto, construido alrededor del procesador 6502. La máquina apareció en 1976 y se llamó PET 2001, un nombre amistoso (*pet* en inglés significa "animal de casa") que se escogió para que no se espantara el comprador de la calle considerándolo altamente técnico.

Pero mientras el PET entraba en el mercado, otros dos innovadores se preparaban en un garaje de California para comercializar su ordenador. Steve Wozniak (véase p. 155) siempre había deseado tener un ordenador, y al hacerse socio del Homebrew Computer Club comprobó que lo podía hacer. Diseñó un ordenador en un único tablero de

circuitos y, con su amigo Steve Jobs, empezó a fabricarlo y venderlo. A su tablero lo llamaron el Apple I. Alojado en una caja con teclado, el ordenador finalmente se transformaría en el clamoroso Apple II. Esta máquina salió justo después del PET de Peddle e incubó toda una industria casera de fabricación de software y hardware.

La Tandy Corporation de Fort Worth (Texas) tenía sus propias ideas para el mercado del pequeño ordenador. La corporación fabricaba, y continúa haciéndolo, una amplia gama de artículos electrónicos, como equipos de alta fidelidad, sintetizadores y radios, vendiéndolos a través de su cadena de tiendas. El ordenador personal representaba una ampliación natural de su línea comercial, y a través de las tiendas Radio Shack ya contaba con una red de distribución para todo Estados Unidos. El resultado fue el TRS-80 Model 1, otro gran éxito en el mercado norteamericano. TRS corresponde a las siglas de Tandy Radio Shack, y el 80 alude al microprocesador utilizado, el Zilog Z80. Zilog era otra nueva firma de chips que había producido un procesador similar al Intel 8080 pero con sustanciales mejoras.

Al contar el TRS-80 Model 1 con un microprocesador Z80, y el Apple II y el Commodore PET con sendos 6502, los ordenadores personales comenzaron a diversificarse en el hardware. Pero junto con esta primera elección trascendental del tipo de consumidor, llegaron los problemas asociados con incompatibilidad de máquinas y software no estandarizado. La clase de microprocesador utilizado en las primeras máquinas es significativa, porque el chip determina la elección del software que terceras partes comercializan en el mercado. Mientras se desarrollaba el hardware, se estaban estableciendo, asimismo, los estándares relativos al software.

En 1972, un joven llamado Gary Kildall era asesor de Intel. Su firma, Microprocessor Application Associates, estaba trabajando en un lenguaje para ordenador que los ingenieros de Intel pudieran utilizar para escribir software destinado a los nuevos chips de microprocesador fabricados por Intel. Kildall pensó que era posible conectar un microprocesador con memoria a una unidad de disco flexible de 8 pulgadas y a un teletipo, con el fin de proporcionarle a cada ingeniero un ordenador propio. Pero Intel prefirió continuar con su práctica de que todos sus ingenieros compartieran una máquina de unidad principal.

El alma de la empresa
Steve Wozniak diseñó y construyó, en el garaje de su casa, el primer Apple I (un TCI sin carcasa). Cuando se modificó el diseño y se lo colocó en una caja, lo que dio lugar a la creación del Apple II, su amigo Steve Jobs hizo del Apple el éxito comercial que es actualmente

Steve Jobs

Cortesía de Apple

Kildall y su amigo John Torode, en otro garaje californiano, montaron ellos solos un sistema. Torode construyó el hardware para hacer que el disco flexible funcionara con el procesador, y Kildall escribió el software en virtud del cual el procesador podía manejar el disco. Al programa se lo llamó CP/M (*Control Program/Microcomputers*), nombre derivado del trabajo de Kildall con el lenguaje de programación de Intel, al que se le había dado la denominación de PL/M (*Programming Language/Microcomputers*).

El primer sistema operativo en disco para micros lo asumieron rápidamente los fabricantes de hardware que deseaban dotar a sus máquinas de unidades de disco. También el software influyó en su diseño: el CP/M sólo operaba en el 8080 y en los procesadores más rápidos 8085 de Intel, así como en el similar Z80 de Zilog. El Z80 se convirtió en el chip estándar para cualquier máquina CP/M, y la compatibilidad con el CP/M era la obsesión de las firmas de software.

Aparte de los sistemas operativos, los ordenadores personales necesitaban un lenguaje de programación en el cual la gente pudiera escribir sus programas. El BASIC, desarrollado en el Dartmouth College (Estados Unidos) como un lenguaje fácil de aprender, era la alternativa obvia.

Bill Gates, graduado en Seattle, ideó un intérprete de BASIC para micros, un programa de traduc-


Cortesía de Apple

Gary Kildall
Los sistemas operativos más recientes los desarrollan grandes equipos de programadores, pero el CP/M lo escribió íntegramente Gary Kildall. Incluso algunas de las últimas versiones reflejaban el hecho de que lo compuso para un hardware muy imperfecto

Steve Wozniak



Cortesía de Osborne

Adam Osborne
Algunas personas dicen de él que es un "cazador furtivo convertido en guardabosques"; Adam Osborne fue durante muchos años un periodista especializado en artículos sobre microordenadores, antes de que creara su propia empresa y produjera el primer ordenador portátil del mundo

ción que cabía en un chip de memoria limitada y que se podía incorporar a una máquina personal. Microsoft, la empresa de Gates, se convirtió en la productora estándar de lenguajes, así como Digital Research se erigió en la productora estándar de sistemas operativos, lo que reportó una fortuna tanto a una como a otra.

A estos progresos siguieron rápidos avances en hardware y en software de aplicaciones. Dan Bricklin y Bob Frankston produjeron el primer programa de hoja electrónica para micros, el VisiCalc, en su empresa Software Arts. Distribuido por Personal Software para el Apple II, éste se convirtió en el paquete de aplicaciones más vendido de todos los tiempos, y para denotar con mayor claridad su relación, Personal Software cambió su nombre por VisiCorp. MicroPro, de Seymour Rubinstein, produjo el WordStar, que se convirtió en el mayor éxito de ventas del mercado de tratamiento de textos CP/M.

El hardware en el que se ejecutaban estos paquetes se abarató y se hizo más potente. Adam Osborne, que se inició como escritor técnico, periodista y editor de software, después de trasladarse de Gran Bretaña a Estados Unidos, lanzó un celebrado ordenador de gestión empresarial que en su precio, ya de por sí competitivo, incluía una gran cantidad

Sir Clive Sinclair
Siguiendo a productos suyos tan innovadores como equipos de alta fidelidad, calculadoras, radios en miniatura, minitelevisores y relojes digitales, el éxito sin precedentes de sus microordenadores (ZX80, ZX81 y el Spectrum) le valió el título de *sir* en 1983

Cortesía de Sinclair Research

Cortesía de Sinclair Research

de software caro. Y está, por supuesto, sir Clive Sinclair, que estableció nuevos niveles de precio con el ZX80, ZX81, y ZX Spectrum, poniendo los ordenadores personales al alcance de millones de usuarios noveles.

En los dos últimos años, el estándar del microordenador lo ha establecido la IBM con el IBM PC. Lanzada al mercado en 1982, la popularidad de esta máquina aumenta día a día. Virtualmente todas las casas de software y fabricantes de periféricos de hardware en la actualidad están produciendo material para el PC, lo cual, a su vez, es un factor de peso para que cada vez más personas opten por esta máquina.

El IBM PC ha reunido a varios de los pioneros de los primeros tiempos de la industria del micro. El microprocesador proviene de Intel, que fue la iniciadora de esa tecnología; el sistema operativo es de la Microsoft de Bill Gates, diversificadora de lenguajes, y de la Digital Research de Gary Kildall, y dos de los primeros paquetes de software con que contó la máquina fueron el VisiCalc y el WordStar.

Steve Wozniak y Steve Jobs dirigen todavía la Apple, que en líneas generales es la competidora directa de la IBM, y ha apostado todas las expectativas de su empresa en favor de la revolucionaria tecnología del Lisa (véase p. 261) y el Macintosh (una versión reducida del Lisa a un precio algo más asequible). Por su parte, Chuck Peddle fundó su propia compañía, la Sirius, y se apropió de un gran sector del mercado británico antes de que la poderosa IBM se implantara en el Reino Unido; desde entonces, la empresa fundada por este pionero ha atravesado dificultades financieras.

Pero seguramente Peddle saldrá adelante. La corta historia del negocio del micro demuestra que los iniciadores son también los supervivientes, incluso aunque las multinacionales intenten ganar la partida.



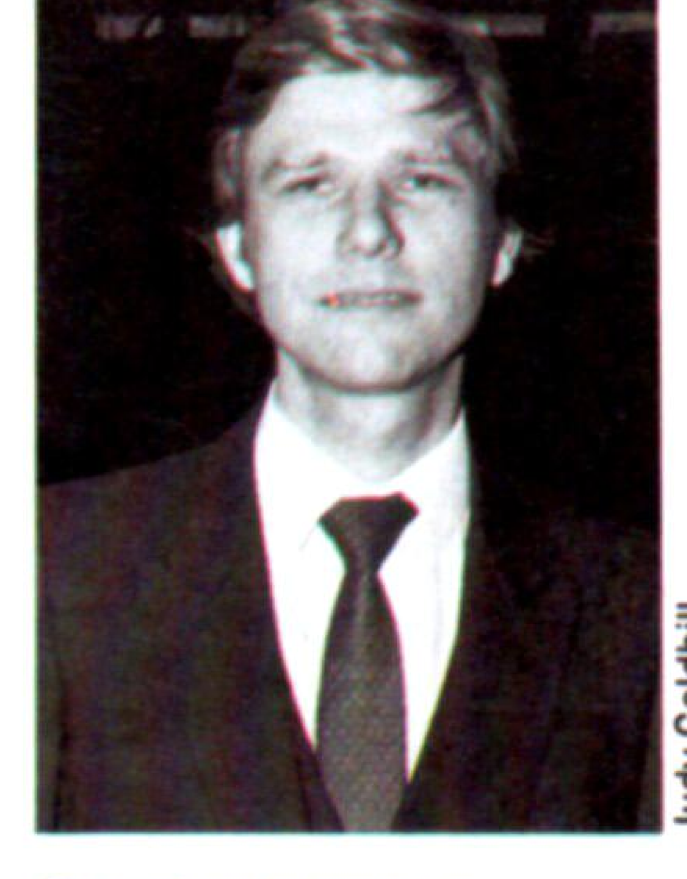
Cortesía de Hewlett Packard

Comienzos menudos
Sorprendentemente, la tecnología de los microordenadores se desarrolló más a partir de las sofisticadas calculadoras programables (como esta Hewlett-Packard HP65) que de la anterior generación de miniordenadores

Ian McKinnell

Una de las grandes
La aceptación por parte de la IBM de la viabilidad del microordenador no tuvo lugar hasta 1982, pero aun así tuvo el efecto previsible. En la actualidad, casi todos los nuevos microordenadores de gestión alardean de ser compatibles con el IBM PC para capitalizar la inmensa base de software existente

Herman Hauser



Judy Goldhill

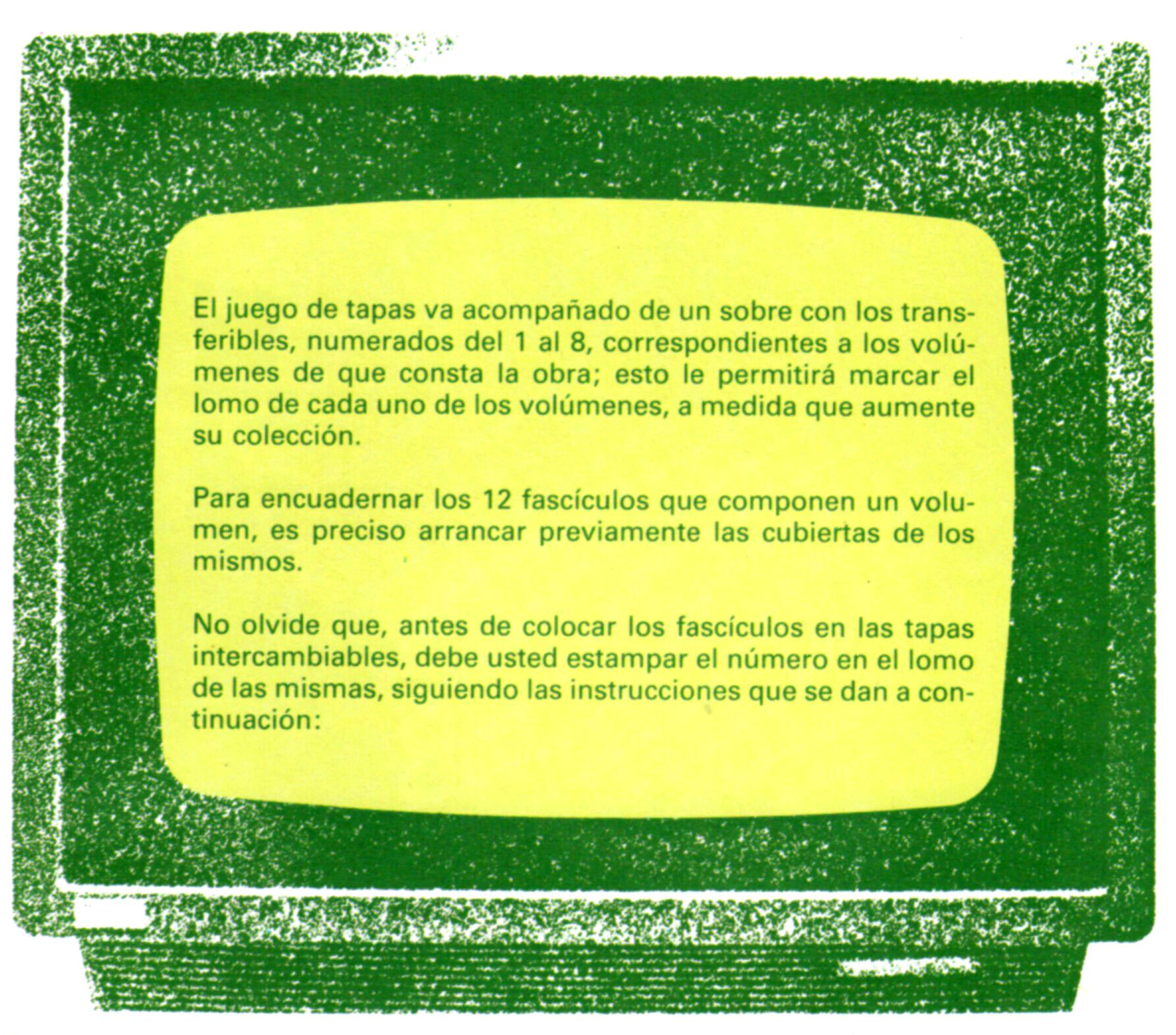
Fecunda colaboración
Aunque menos innovadora en cuanto a precio que Sinclair, la contribución de Chris Curry y Herman Hauser (como diseñadores y directores de los ordenadores Acorn) ha sido igualmente valiosa. Tanto el Acorn Atom como el BBC Microcomputer y el Electron son hitos por derecho propio

Judy Goldhill

Chris Curry



Con este fascículo se han puesto a la venta las tapas correspondientes al segundo volumen.



El juego de tapas va acompañado de un sobre con los transferibles, numerados del 1 al 8, correspondientes a los volúmenes de que consta la obra; esto le permitirá marcar el lomo de cada uno de los volúmenes, a medida que aumente su colección.

Para encuadernar los 12 fascículos que componen un volumen, es preciso arrancar previamente las cubiertas de los mismos.

No olvide que, antes de colocar los fascículos en las tapas intercambiables, debe usted estampar el número en el lomo de las mismas, siguiendo las instrucciones que se dan a continuación:



1. Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.
2. Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.
3. Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.

Editorial Delta

Cada sobre de transferibles contiene una serie completa de números, del 1 al 8, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.

Ya están a su disposición, en todos los quioscos y librerías, las tapas intercambiables para encuadernar 12 fascículos de

mi COMPUTER

Cada juego de tapas va acompañado de una colección de transferibles, para que usted mismo pueda colocar en cada lomo el número de tomo que corresponda





